TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CD. JUÁREZ

Informe técnico de residencia profesional

Diseño de prototipado y fabricación de brazos robóticos didácticos a partir de impresión 3D.

QUE PRESENTA:

Diego Alberto Delgado Saenz

Idaly Fernandez Esquivel

PARA OBTENER EL TITULO DE:

Ingeniero Electromecanico

Ingeniera Mecatronica

Cd. Júarez, Chih. Noviembre 2025

AGRADECIMIENTOS

Agradezco a Dios por haberme permitido llegar hasta este momento y darme la fortaleza necesaria para culminar esta etapa.

A mi familia, por su apoyo constante, su paciencia y por creer en mi incluso en los días momentos más difíciles.

A mis profesores y compañeros, quienes con su ejemplo, conocimiento y compañerismo me inspiraron a dar lo mejor de mí y a convertirme en una mejor versión de mí mismo.

DEDICATORIA

A mis papás, porque con su amor, paciencia y ejemplo me enseñaron a ser fuerte, por inculcarme el respeto, la perseverancia y la visión para ir siempre un paso más allá. Gracias por impulsarme a mirar más lejos y nunca conformarme con menos de lo que puedo ser.

A mis profesores y compañeros, por contribuir a que mi paso por la universidad fuera una experiencia significativa, enriquecedora y llena de aprendizaje, acompañamiento y crecimiento personal.

A Dios, por caminar conmigo en cada etapa de este viaje. Porque cuando mis fuerzas se agotaron, Él me sostuvo. Gracias por ser mi refugio, mi compañía y mi luz en cada decisión y cada desafío.

RESUMEN

En el contexto de la educación tecnológica, los brazos robóticos representan una herramienta poderosa para acercar a los estudiantes al mundo de la robótica, la programación y la automatización. Este proyecto propone el desarrollo de un brazo robótico didáctico de cinco grados de libertad, diseñado para ser reproducible, económico y robusto, ideal para su implementación en entornos educativos.

El sistema fue construido mediante impresión 3D con filamento PETG, utilizando servomotores de distintas capacidades según la demanda de cada articulación. El control del movimiento se logra mediante un microcontrolador Arduino Nano, mientras que la interfaz gráfica, desarrollada en Visual Studio y LabVIEW, permite una interacción intuitiva por parte del usuario.

A través de este desarrollo, se busca facilitar el aprendizaje práctico de principios mecatrónicos, brindando a estudiantes una plataforma real que integra diseño mecánico, electrónica y programación en un solo dispositivo accesible.

CONTENIDO

AGRADECIMIENTOS....2

DEDICATORIA....3

RESUMEN....4

1. INTRODUCCIÓN....7
Planteamiento del problema....8
1.1 Antecedentes....8
1.2 Descripción del problema....10
1.3 Preguntas de Investigación....11
1.4 Hipótesis....12
1.5 Objetivos....12
1.6 Justificación....13
1.7 Delimitación....14

2. MARCO TEÓRICO....15
2.1 Definición de brazo robótico....15
2.2 Clasificación de brazos robóticos....15
2.3 Grados de libertad....18
2.4 Estudio del modelado y la cinemática del robot....19
2.5 Cinemática Directa en Robótica....20
2.6 Cinemática Inversa en Robótica....22
2.7 Polimeros y plásticos....24
2.8 Impresión 3D....25
2.9 Materiales de Impresión 3D....30
2.10 Servos....31
2.11 La modulación por ancho de pulso y su aplicación en Servomotores....35
2.12 Lab View....37
2.13 Arduino....39

3. DESARROLLO....41
3.1 Impresión del modelo 3d inicial....43
3.2 Postprocesado de las piezas....**Error! Bookmark not defined.**
3.3 Modelado y análisis teórico....49
3.4 Arquitectura del sistema de control electrónico....62
3.5 Firmware inicial del control de movimiento....67

3.6 Armado y cableado del prototipo.....74
3.7 Rediseño del modelo.....76
3.8 Recuento de materiales.....79
3.9 Desarrollo de interfaz gráfica en labview.....83
3.10 Validación y pruebas funcionales.....88
3.11 Evaluación del desempeño del brazo robótico.....89

CONCLUSIONES.....90

REFERENCIAS BIBLIOGRÁFICAS.....92

ANEXOS.....95
ANEXO 1. CRONOGRAMA PRELIMINAR DE ACTIVIDADES.....95
ANEXO 2. ANÁLISIS DE ESFUERZOS O ESTÁTICO EN SOLIDWORKS (ESCALA FACTOR DE SEGURIDAD).....96
ANEXO 3. DIAGRAMA ELECTRÓNICO DE CONEXIONES.....100
ANEXO 4. CÓDIGO DE FIRMWARE DE ARDUINO: Lifa Base para Lab View.....101
ANEXO 5. REDISEÑOS POR PIEZA.....104
ANEXO 6. LISTA DE MATERIALES (BOM).....107
ANEXO 7. CAPTURAS DEL PROGRAMA EN LABVIEW.....108
ANEXO 8. Programa del Nodo de Fórmula en Labview para los cálculos de cinemática directa e inversa del robot.....135

LISTA DE TABLAS

Tabla 1. Software de modelado.....27
Tabla 2. Tabla de Denavit-Hartenberg para el brazo robótico de estudio.....51
Tabla 3. Cálculos de consumo de corriente eléctrica de los motores del robot.....64

LISTA DE FIGURAS

Figura 1. Robot Industrial.....15
Figura 2. Brazo Robótico KUKA Modelo.....16
Figura 3. Tipos de configuraciones.....17
Figura 4. Descripción grafica de articulación tipo prismática.....17
Figura 5. Ejemplo de articulaciones en un cobot industrial.....18
Figura 6. Composición molecular de los polímeros.....24
Figura 7. Compuesto de petróleo del plástico.....24
Figura 8. Objetos de plástico.....25
Figura 9 y 10. Maquinado Industrial.....26
Figura 11. Software de modelado.....27
Figura 12. Proceso de Impresión 3D.....28
Figura 13. Dimensionado del objeto de impresión.....28
Figura 14. Reconstrucción para la impresión.....29
Figuras 15 y 16. Post procesado de la pieza impresa.....29
Figura 17. Diagrama esquemático de funcionamiento de un servo.....31
Figura 18. Servo Motor MG995.....32
Figura 19. Servo motor MOT-110.....33
Figura 20. Componentes de un servomotor.....35
Figura 21 y 22. Modulación de ancho de pulso. Fuente: http://www.forosdeelectronica.com/.....35
Figura 23. Ejemplificación del PWM. Fuente: http://cfievalladolid2.net.....37
Figuras 24 y 25. Ventana de Front Panel y Block Diagram en Lab View.....39
Figura 26. Placa de Arduino NANO.....40
Figura 27. Programa de Fusion 360.....41
Figura 28. Modelo 3D del brazo robótico.....42
Figura 29. Apariencia interna promedio de una pieza conforme a su porcentaje de relleno.....44
Figura 30. Diferencia de solidez de una impresión 3D según el número de capas superficiales.....45
Figuras 31 y 32 . Nivel de detalle de la impresión respecto a altura de capa.....46
Figura 33. Selección de impresora en Creality.....46
Figura 34. Organizado de piezas por bandejas para un equilibrio de tiempo de impresión.....47
Figuras 35, 36, 37, 38. Especificaciones de impresión por bandeja.....47
Figura 39. Posición de origen del robot en vista isométrica de Fusión 360.....49
Figura 40. Morfología del robot.....50
Figura 41. Propiedades físicas del modelo según el material de PETG.....59
Figura 42. Configuración del esfuerzo aplicado del motor para el análisis.....60
Figura 43. Resultado análisis de esfuerzos.....60
Figura 44. Resultado análisis de desplazamientos.....61
Figuras 45 y 46.Icono del VI Package Managerx.....68
Figura 47. Digilent LINX extension.....68
Figura 48. Ubicación de la extensión dentro de Lab View.....69
Figura 49. Menú Principal de LINX Firmware Wizard.....70

Figura 50. Selección de puerto disponible.....70
Figuras 51 y 52. Seguimiento de comunicación entre Lab View y el microcontrolador. 71
Figura 53. Respuesta satisfactoria del microcontrolador.....71
Figura 54. Diseño inicial del brazo.....76
Figura 55. Diseño final del brazo.....76
Figura 56. Ensamble con los modelos finales del robot.....77
Figura 57. Bandeja de impresión 1.....79
Figura 58. Bandeja de impresión 2.....80
Figura 59. Bandeja de impresión 3.....80
Figura 60. Bandeja de impresión 4.....81
Figura 61. Bandeja de impresión 5.....81
Figura 62. Bandeja de impresión 6.....82
Figura 63. Descripción de costos en materiales para un brazo robótico.....82
Figura 64. Front panel del Menú Principal.....84
Figura 65. Front Panel Movimiento libre.....85
Figura 66. Front Panel de Ecuaciones de posición.....86
Figura 67. Front panel de secuencias.....87

1. INTRODUCCIÓN

La robótica se ha convertido en una disciplina clave dentro de la formación tecnológica actual, al integrar conocimientos de mecánica, electrónica y programación en aplicaciones concretas. En el ámbito educativo, los sistemas robóticos representan una oportunidad única para que los estudiantes desarrollen habilidades prácticas y comprendan de manera más tangible los principios de la ingeniería.

Sin embargo, el acceso a equipos didácticos que permitan este tipo de aprendizaje aún representa un reto en muchas instituciones debido a los altos costos, la complejidad técnica o la falta de recursos. Ante esta situación, surge la necesidad de desarrollar herramientas accesibles, funcionales y replicables que puedan ser utilizadas como apoyo en la enseñanza de la robótica y la automatización.

Este proyecto aborda dicha necesidad mediante el diseño y construcción de un brazo robótico didáctico de cinco grados de libertad, fabricado con impresión 3D y controlado por un microcontrolador Arduino Nano. Su diseño modular y su interfaz gráfica permiten que el sistema sea utilizado por estudiantes de distintos niveles académicos para explorar conceptos clave en robótica, desde el movimiento articulado hasta la programación de trayectorias.

Planteamiento del problema.

A partir del contexto descrito, se presenta a continuación un análisis detallado del problema identificado, los antecedentes relevantes, la justificación del proyecto y los objetivos que guían su desarrollo.

1.1 Antecedentes

En los últimos años, la robótica educativa ha cobrado gran relevancia como herramienta para la enseñanza de conceptos de ingeniería, programación y automatización. Los brazos robóticos, en particular, se han convertido en un recurso didáctico clave debido a su versatilidad y capacidad para simular tareas industriales y de manipulación de objetos, facilitando así el desarrollo de habilidades en los estudiantes.

Desde la década de 1980, los primeros manipuladores robóticos comenzaron a utilizarse en entornos académicos para la enseñanza de cinemática y control, destacando modelos como el PUMA 560, desarrollado por *Unimation*. Con el tiempo, surgieron plataformas más accesibles, como el *Lynxmotion AL5D* y el *TurtleBot Arm*, diseñadas para la enseñanza de robótica y automatización. La llegada de microcontroladores como Arduino y Raspberry Pi permitió que la programación y el control de estos sistemas fueran más intuitivos y accesibles, facilitando su adopción en entornos educativos.

A pesar de estos avances, muchas soluciones actuales siguen enfocadas en aplicaciones industriales o presentan costos elevados, lo que limita su implementación en instituciones con recursos limitados. Este proyecto busca abordar esa necesidad mediante el diseño y construcción de 10 brazos robóticos didácticos de 5 grados de libertad, empleando tecnologías de bajo costo y software intuitivo para su control.

La evolución de la impresión 3D ha sido un factor clave en la accesibilidad y desarrollo de la robótica educativa. En 1984, Alain Le Méhauté, Olivier de Witte y Jean-Claude André presentaron una patente para el proceso de

estereolitografía (SLA). Tres semanas después, Charles Hull patentó un proceso similar enfocado en la creación de prototipos, lo que lo llevó a fundar *3D Systems* en 1986, empresa pionera en la impresión 3D a nivel industrial. Su tecnología utilizaba un láser ultravioleta para solidificar capas de resina acrílica, repitiendo el proceso hasta formar un objeto tridimensional. Para 1987, el prototipo rápido (*Rapid Prototyping*) ya era una realidad comercial.

Entre 1989 y 1990, Scott Crump, fundador de *Stratasys*, desarrolló la técnica de *Fused Deposition Modeling* (FDM), basada en la deposición de capas sucesivas de material fundido que se solidifican para formar objetos tridimensionales. Esta innovación redujo costos y democratizó el acceso a la impresión 3D, permitiendo su adopción en sectores no industriales, como la educación y el diseño de prototipos.

Posteriormente, en 2006, se introdujo la primera máquina de *Selective Laser Sintering* (SLS), que emplea un láser para fusionar partículas de material y construir objetos tridimensionales. Este avance impulsó la personalización masiva y la fabricación bajo demanda de piezas industriales, consolidando la impresión 3D como una herramienta clave en diversas aplicaciones, incluida la robótica educativa.

1.2 Descripción del problema

Actualmente, existe una falta de herramientas didácticas accesibles y económicas que permitan a los estudiantes y profesionales en formación experimentar con sistemas robóticos complejos. Los brazos robóticos disponibles en el mercado suelen ser costosos y están diseñados principalmente para aplicaciones industriales, lo que limita su uso en entornos educativos.

Además, la falta de integración entre hardware y software amigable dificulta el aprendizaje de conceptos clave como la cinemática, el control de movimientos y la programación de este.

Este proyecto aborda estos problemas mediante el desarrollo de un sistema robótico didáctico de bajo costo, basado en impresión 3D, servomotores y control del robot por un microcontrolador. Aunado a esto, la interfaz de control desarrollada en Visual Studio y LabVIEW permitirá a los usuarios programar y manipular los movimientos del brazo de manera intuitiva, facilitando el aprendizaje y la experimentación.

1.3 Preguntas de Investigación

1. ¿Qué características debe tener un brazo robótico didáctico para ser efectivo en entornos educativos de nivel medio superior y superior?

Un brazo robótico didáctico efectivo debe ser de bajo costo, fácil de ensamblar, seguro para su uso en aulas, y permitir la integración con plataformas de programación accesibles como Arduino. Además, debe ser lo suficientemente versátil para enseñar conceptos de mecánica, electrónica y programación.

2. ¿Cómo influye la elección de materiales, como el PETG, en la funcionalidad y durabilidad de un brazo robótico impreso en 3D para uso educativo?

El uso de PETG en la impresión 3D de componentes robóticos ofrece una mayor resistencia mecánica y térmica en comparación con otros materiales como el PLA, lo que resulta en una mayor durabilidad y fiabilidad del brazo robótico en entornos educativos donde el uso frecuente es común.

3. ¿Qué ventajas ofrece la integración de microcontroladores como el Arduino Nano en el control de brazos robóticos educativos?

El Arduino Nano proporciona una plataforma compacta y económica que facilita la programación y el control de servomotores en brazos robóticos. Su compatibilidad con diversas interfaces de software y su amplia documentación lo hacen ideal para aplicaciones educativas.

4. ¿De qué manera las interfaces gráficas desarrolladas en plataformas como Visual Studio y LabVIEW mejoran la interacción de los estudiantes con sistemas robóticos?

Las interfaces gráficas desarrolladas en Visual Studio y LabVIEW permiten a los estudiantes interactuar de manera más intuitiva con los sistemas robóticos, facilitando la comprensión de conceptos complejos mediante visualizaciones y controles accesibles, lo que mejora el proceso de aprendizaje.

5. ¿Qué impacto tiene el diseño modular en la adaptabilidad y escalabilidad de brazos robóticos educativos?

Un diseño modular permite que los brazos robóticos sean fácilmente adaptables y escalables, facilitando la personalización según las necesidades educativas específicas y permitiendo la incorporación de nuevas funcionalidades o la reparación de componentes individuales sin necesidad de reemplazar todo el sistema.

1.4 Hipótesis

Un brazo robótico de cinco grados de libertad, construido con PETG mediante impresión 3D y controlado con un Arduino Nano, permitirá ejecutar movimientos precisos y coordinados que simulan procesos industriales, siendo adecuado para su uso didáctico.

1.5 Objetivos

Objetivo General:

Desarrollar un brazo robótico didáctico de 5 grados de libertad, utilizando impresión 3D y servomotores controlados por un microcontrolador,

con una interfaz de control desarrollada en Visual Studio y LabVIEW, para facilitar el aprendizaje de conceptos de robótica y automatización en entornos educativos.

Objetivos Específicos

- Adecuar el modelo 3D para garantizar su compatibilidad con la impresión 3D y asegurar la resistencia de las piezas.
- Seleccionar y configurar los componentes electrónicos correctos como servomotores y el microcontrolador.
- Desarrollar el control de los motores programando el microcontrolador para movimientos precisos y coordinados.
- Crear la interfaz gráfica en Visual Studio o LabVIEW, asegurando su facilidad de uso.
- Realizar calibración y ajustes para garantizar movimientos precisos y óptimos.

1.6 Justificación

Este proyecto tiene como finalidad contribuir al campo de la robótica educativa mediante el desarrollo de una herramienta didáctica asequible y fácil de utilizar. La implementación de tecnologías como la impresión 3D y microcontroladores permite reducir costos sin comprometer la funcionalidad, lo que facilita su adopción en instituciones educativas con recursos limitados.

La integración de tecnologías como la impresión 3D, microcontroladores avanzados y software de control ha permitido reducir costos y aumentar la accesibilidad en el desarrollo del proyecto.

Además, la integración de una interfaz gráfica desarrollada en Visual Studio y LabVIEW simplifica el proceso de aprendizaje, permitiendo a los usuarios enfocarse en conceptos clave como la programación, el control de movimientos y la cinemática de robots. Este proyecto no solo beneficiará a estudiantes y profesores, sino que también servirá como base para futuros desarrollos en el área de la robótica educativa.

La diferencia entre la fabricación tradicional y la impresión 3D es como se forman los objetos, a diferencia de la impresión 3D, en la fabricación tradicional involucra una combinación de trituración, formación, flexión, fundición, corte, etc., para después montarlos. En el proceso de dar forma al material se pierde material, así como energía utilizada por las máquinas especializadas, que únicamente cumplen con la función de producir ese objeto y no más, esto conlleva a un alto costo en piezas de geometría compleja, eso sin mencionar el tiempo de espera por las mismas.

Por el contrario, una impresora 3D puede producir una pieza simple en una sola operación e incluso salir completamente montada, si cuenta con partes móviles, después de un trabajo de limpieza y acabado simple, estaría listo para su uso. El uso de esta tecnología, que ha pasado de ser exclusivamente de uso industrial a ser un proceso disponible para el público en general, reduce el tiempo de generación de prototipos de prueba, lo que genera menores costos ya que la materia prima es más barata y reutilizable.

1.7 Delimitación

A partir del planteamiento del problema, se puede definir de manera formal el proyecto y entender las limitaciones que acompañarán su desarrollo. En cuanto al alcance tecnológico, el proyecto se centrará en el diseño de un brazo robótico de 5 grados de libertad, utilizando servomotores y motores paso a paso como actuadores principales. En cuanto al hardware, se empleará un microcontrolador para el control de los motores y la

comunicación con la interfaz de usuario. Para el desarrollo del software, la interfaz de control se programará en Visual Studio y LabVIEW, sin incorporar otras plataformas de programación. El entorno de aplicación estará limitado a uso didáctico en laboratorios educativos, sin contemplar aplicaciones industriales o de alta precisión.

2. MARCO TEÓRICO.

2.1 Definición de brazo robótico

Una de las ramas más visibles de la automatización es la robótica. Según “Robot Institute of America” (RIA), un robot es “un manipulador reprogramable, multifuncional, diseñado para mover materiales, partes, herramientas y objetos especiales a través de movimientos programados variables para el cumplimiento de una variedad de tareas”.

Figura 1. Robot Industrial

2.2 Clasificación de brazos robóticos

Los robots se pueden clasificar en dos grupos:

a) Máquinas capaces de realizar trabajos automáticos de modo continuo o aleatorio, con un cierto grado de autonomía, como robots industriales, militares, de limpieza, etc.

b) Máquinas cuya apariencia y funciones pretenden imitar las de personas o animales, lo que incluye también los trajes a modo de exoesqueleto y dispositivos que aumentan la fuerza, entre otros.

Las clasificaciones más importantes que se pueden realizar de los robots atienden a la arquitectura con que se construyen y la configuración física o anatomía de estos. La arquitectura de un robot es la forma que tienen los mecanismos y cómo están configurados. Entre los tipos de arquitectura de robots tenemos poli articulados, móviles, androides, zoomórficos e híbridos.

Nuestro brazo robot es un robot tipo poli articulado, cuya forma y configuración característica es que son básicamente sedentarios y su estructura consiste en mover sus elementos terminales en un número determinado de grados de libertad y se pueden movilizar en uno o más ejes de un sistema de coordenadas. Dentro de esta categoría se encuentran los manipuladores, los robots cartesianos y algunos robots industriales, como el

KUKA.

Figura 2. Brazo Robótico KUKA Modelo.

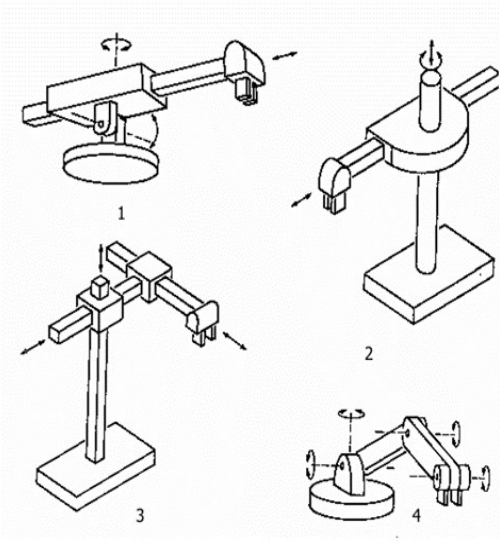
Al hablar sobre la anatomía del robot, nos referimos a la construcción física del robot, los movimientos relativos entre los diversos componentes del cuerpo, brazo y muñeca son proporcionados por una serie de articulaciones. El efector final del robot no se considera como parte de la anatomía del robot.

Existen cuatro configuraciones comunes de robots, donde cada una se diferencia por el tipo de movimientos que se pueden realizar;

- Configuración polar
- Configuración cilíndrica
- Configuración cartesiana
- Configuración brazo articulado (Configuración del brazo de este proyecto)

El brazo robot consiste en un robot manipulador de cadena cinemática abierta. Estos robots constan de dos elementos fundamentales: eslabones y articulaciones. Los eslabones son los elementos mecánicos que constituyen la estructura del robot.

Por medio de las articulaciones o uniones los eslabones pueden enlazarse entre sí para constituir la cadena cinemática del robot, que puede



FUENTE: Groover, Mikell P., et. al. ROBÓTICA INDUSTRIAL. Tecnología, programación y aplicaciones. Pág. 23

ser abierta o cerrada, el robot de este proyecto ya que el último eslabón no se conecta a otro elemento.

Figura 3. Tipos de configuraciones.

Los eslabones presentan un movimiento relativo entre sí por medio de las articulaciones que los interconectan. Las articulaciones pueden ser de varios tipos, aunque las más comunes son las rotacionales (rotativas) o traslacionales (prismáticas).

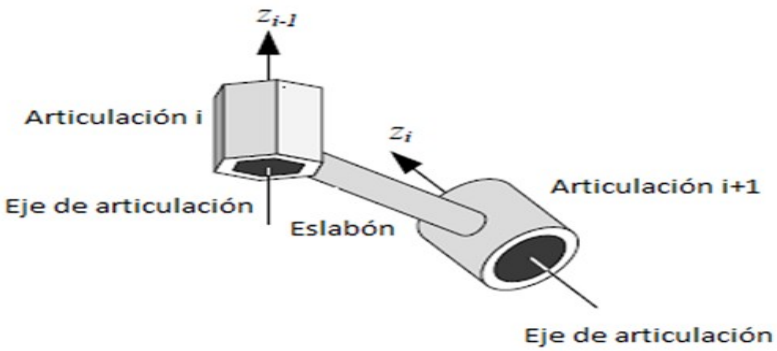


Figura 4. Descripción grafica de articulación tipo prismática.

2.3 Grados de libertad

Un concepto importante para el robot es el de grados de libertad (GDL, o DOF – Degrees Of Freedom en inglés), que se refieren al conjunto de variables independientes que permiten determinar la posición y orientación del elemento final del robot. Los eslabones de un robot se mueven por medio de dispositivos denominados actuadores. Dichos actuadores pueden ser hidráulicos, neumáticos o eléctricos.

Los actuadores se encuentran acoplados con las articulaciones del robot. Si las articulaciones son independientes, con una única dirección de movimiento o rotación respecto a un eje determinado, el número de articulaciones con que cuenta un manipulador coincide con el número de grados de libertad del manipulador.

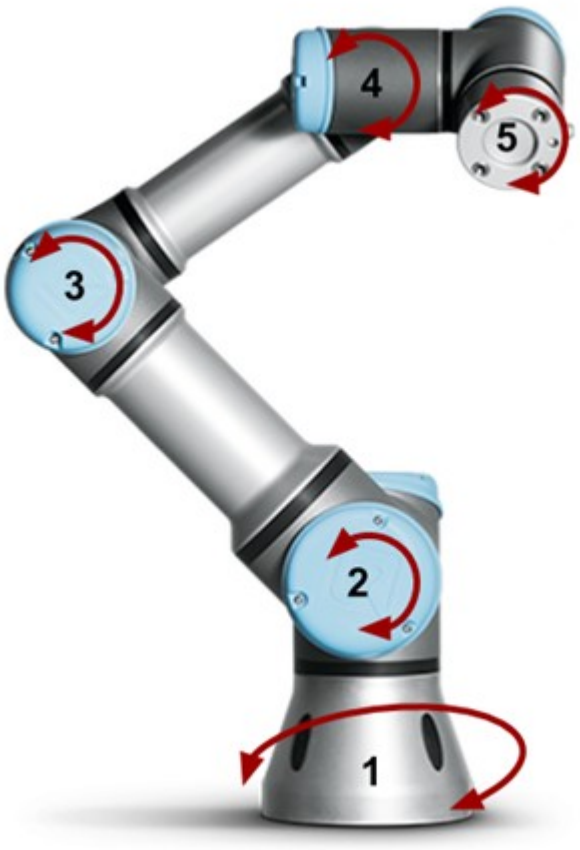


Figura 5. Ejemplo de articulaciones en un cobot industrial.

2.4 Estudio del modelado y la cinemática del robot

Modelar equivale a representar un objeto o sistema. En el caso de los sistemas mecánicos, como el robot articulado de cinco grados de libertad que se analiza en este trabajo, el modelo requerido se enfoca principalmente en describir su movimiento.

El propósito de este trabajo es emplear un modelo de posición que permita estudiar el comportamiento cinemático del robot articulado. Para ello, se utilizarán las ecuaciones de posición que faciliten la determinación de las coordenadas de un punto de interés, partiendo desde un origen definido

en un sistema de coordenadas cartesianas.

El interés principal radica en modelar las rotaciones de cada uno de los eslabones que conforman el robot, con el objetivo de abordar dos problemas fundamentales de la cinemática: el problema directo y el problema inverso, enfocados específicamente en el posicionamiento.

Problema cinemático inverso:

Este problema consiste en calcular los ángulos de rotación que deben adoptar los motores que accionan los eslabones del robot, con base en una posición conocida del efector final (el punto "pot"). De forma general, se formula como:

“Dadas las coordenadas (Xpot, Ypot, Zpot), determine los ángulos de las articulaciones del robot necesarios para alcanzar dicha posición.”

Problema cinemático directo:

Este problema busca obtener la posición del efector final (el punto "pot") a partir de los valores conocidos de los ángulos de cada articulación. Se plantea de la siguiente manera:

“Dadas las rotaciones de los cinco eslabones que conforman el robot, determine las coordenadas (Xpot, Ypot, Zpot) del efector final.”

2.5 Cinemática Directa en Robótica

En otras palabras, se trata de una función que describe cómo se mueve la “mano” del robot, en función de los movimientos de sus “articulaciones”. Esta información es esencial para la operación, programación y control de robots, y constituye el primer paso para entender cómo se representa el

movimiento mecánico en el espacio tridimensional.

En un robot con múltiples grados de libertad (GDL), cada articulación puede girar o desplazarse. Conocidos los valores de estas variables (denominadas coordenadas articulares), el problema cinemático directo consiste en determinar las coordenadas cartesianas del extremo del robot, así como su orientación espacial. Esta relación se representa usualmente mediante ecuaciones matemáticas no lineales que deben construirse cuidadosamente utilizando métodos de transformación de coordenadas.

Matemáticamente, si se tiene un vector de coordenadas articulares:

$$q = [q_1, q_2, \ldots, q_n]^t$$

El problema cinemático directo busca calcular la posición y orientación del extremo del robot:

$$P = [x, y, z, \varphi, \theta, \psi]^t = f(q)$$

donde (x, y, z) representa la posición en el espacio y (φ, θ, ψ) los ángulos de orientación, comúnmente expresados como ángulos de Euler o mediante matrices de rotación.

2.6.1 Métodos para Resolver la Cinemática Directa

Existen dos métodos principales para abordar este problema:

1. Método Geométrico:

Este método es útil en robots con pocos grados de libertad y se basa en principios de trigonometría y geometría clásica. A partir de diagramas del robot, se utilizan las longitudes de los eslabones y los ángulos de las articulaciones para aplicar el teorema del coseno y del seno. Sin embargo, su aplicación se vuelve limitada y compleja conforme aumenta el número de articulaciones, por lo que es adecuado solo para estructuras planas o simples.

2. Método de Transformaciones Homogéneas (Denavit-Hartenberg):

El enfoque más general y sistemático es el uso del algoritmo de Denavit-Hartenberg (D-H). Este método define cuatro parámetros por articulación: θ_i, d_i, a_i, α_i, y con ellos se construyen matrices de transformación homogénea 4 × 4 que relacionan cada eslabón con el siguiente.

Este método consiste en asignar un sistema de coordenadas a cada eslabón del robot y describir su relación con el siguiente eslabón utilizando cuatro parámetros: θ (ángulo de articulación), d (desplazamiento), a (longitud del eslabón) y α (ángulo de torsión). Estos parámetros se organizan en una tabla conocida como tabla de parámetros D-H, que posteriormente permite construir las matrices de transformación homogénea entre eslabones consecutivos.

Cada matriz de transformación es una representación matemática que combina una rotación y una traslación en el espacio tridimensional, permitiendo describir cómo se mueve un eslabón con respecto al anterior. La multiplicación secuencial de estas matrices da como resultado una matriz global de transformación, la cual describe la posición y orientación del efector final (gripper) en función de las variables articulares del robot. A este resultado se le conoce como la ecuación de movimiento del robot y corresponde al análisis de cinemática directa.

Cada matriz tiene la forma:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicando todas las matrices desde la base hasta el efector final se obtiene una única matriz T que describe su ubicación final respecto al sistema de coordenadas fijo:

$$T = A_1^0 \cdot A_2^1 \cdot \ldots \cdot A_n^{n-1}$$

Una vez construido el modelo cinemático directo de un robot, se puede utilizar para simular trayectorias, verificar si una posición es alcanzable, realizar control visual o generar animaciones del movimiento del robot en entornos virtuales.

2.6 Cinemática Inversa en Robótica

El problema de la cinemática inversa es el complemento directo del cinemático directo. En este caso, se conoce la posición y orientación deseadas del efector final del robot y se desea calcular los valores de las articulaciones necesarias para alcanzarlas. Es decir, dadas las coordenadas cartesianas del extremo del robot, se pretende conocer qué combinación de movimientos articulares es necesaria para que el robot se ubique en esa configuración espacial.

Este problema es más complejo que su contraparte directa, ya que puede tener múltiples soluciones (o ninguna), y frecuentemente involucra resolver ecuaciones no lineales acopladas.

Dado un punto objetivo:

$$P = [x, y, z, \varphi, \theta, \psi]^t$$

se busca encontrar:

$$q = [q_1, q_2, \ldots, q_n]^t \quad tal\ que \quad f(q) = P$$

2.6.1 Métodos para Resolver la Cinemática Directa

1. Métodos Geométricos

Son útiles en robots con estructuras simples. Se basan en construir triángulos formados por los eslabones del robot y aplicar relaciones trigonométricas para encontrar los ángulos articulares. Su ventaja es la claridad y velocidad, pero se limita a robots con geometrías planas o muy regulares.

2. Métodos Analíticos

Cuando se dispone del modelo cinemático directo mediante matrices de transformación homogénea, es posible manipular algebraicamente estas matrices para despejar los ángulos articulares. Esta técnica permite una solución cerrada (analítica), pero es aplicable solo si las ecuaciones lo permiten.

3. Métodos Numéricos

Cuando la solución analítica no es posible o práctica, se utilizan algoritmos iterativos que aproximan la solución mediante técnicas como Newton-Raphson, métodos del gradiente o PSO. Estos métodos prueban distintas combinaciones de valores articulares y evalúan qué tan cerca están de alcanzar el objetivo.

Función de error común:

$$f_{error} = \sqrt{((px - x)^2 + (py - y)^2 + (pz - z)^2)}$$

Singularidades y Múltiples Soluciones:

El problema cinemático inverso puede tener múltiples soluciones o ninguna, dependiendo de si el punto deseado está dentro del espacio de trabajo del robot. También existen singularidades, posiciones donde el robot pierde grados de libertad o la solución se vuelve inestable.

Es importante mencionar que este trabajo se enfocará exclusivamente en el estudio del problema cinemático directo e inverso para el robot articulado de cinco grados de libertad.

2.7 Pólimeros y plásticos.

Los polímeros, son un tipo particular de molécula que se caracteriza porque sus unidades se repiten a lo largo de su extensión. Las pequeñas moléculas que se combinan entre sí mediante un proceso químico, llamado reacción de polimerización, para formar el polímero se denominan monómeros. También existen los elastómeros, que son polímeros que poseen elasticidad, se usan para fabricar gomas, mangueras, neumáticos debido a su flexibilidad. También existen las denominadas fibras que poseen una alta elasticidad y baja extensibilidad (esto permite tejer materiales con estabilidad bidimensional)

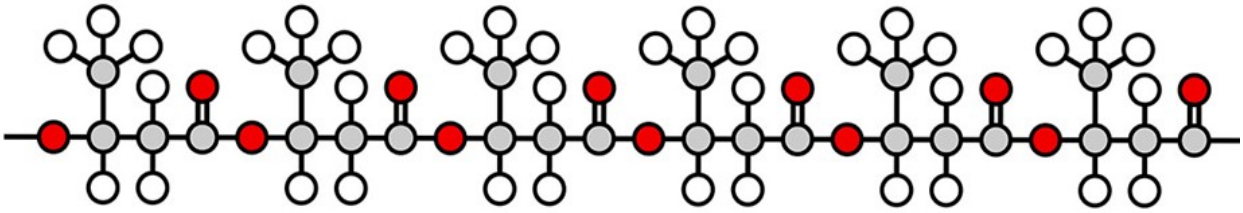


Figura 6. Composición molecular de los polímeros.

Otro tipo de polímero son los plásticos, que se definen como polímeros que proceden de recursos naturales como el petróleo, gas natural, carbón y sal común, que pueden ser moldeados a presión y transformados en diversos objetos con formas diferentes. Sin embargo, el origen de los plásticos no siempre es natural, existen polímeros semisintéticos y sintéticos, mientras que los primeros se obtienen de la transformación de polímeros naturales

(como la nitrocelulosa y caucho vulcanizado), los otros se obtienen industrialmente a partir de monómeros, que son las unidades individuales de los polímeros (nylon, poliestireno, policloruro de vinilo o polietileno).

Figura 7. Compuesto de petróleo del plástico.

En particular los plásticos se dividen en dos grandes grupos debido al comportamiento térmico, los termoestables y los termoplásticos. Los termoestables son plásticos cuya estructura colapsa al calentarse, de modo que no pueden ser reutilizados. Por otro lado, existen los termoplásticos, su principal característica es que cuando se someten a calentamiento empiezan a reblandecerse y a fluir nuevamente, de modo que al bajar la temperatura vuelvan a su estado sólido. Gracias a esta cualidad estos plásticos pueden ser procesados repetidamente de forma indefinida.

Figura 8. Objetos de plástico.

2.8 Impresión 3D

Acerca de la impresión 3D, se considera una tecnología del futuro, pero su funcionamiento parte de los sistemas de tecnología modernos digitales. Partiendo de la automatización, se pasó de la fabricación tradicional hecha de forma manual a través de operadores humanos que manipulaban la acción de las máquinas herramientas (es decir, máquinas que dan forma a un material bruto generalmente por sustracción), a los sistemas CNC (Computer Numeric Control, o Control Numérico Computarizado) el cual es un sistema de automatización que se utiliza para controlar las máquinas herramienta.

Este sistema de automatización de máquinas ha revolucionado la industria gracias a la simplificación de software de diseño en conjunto con los lenguajes de programación que rigen a las maquinas electrónicas que usamos diariamente.

Figura 9 y 10. Maquinado Industrial

El modelado de objetos en tres dimensiones es muy sencillo hoy en día gracias al software CAD, esencialmente es un programa de ordenador que sirve para la creación, edición, análisis y visualización de modelos tridimensionales (SolidWorks, AutoCAD, Inventor, entre los más destacados). No solo sirven para hacer visualizaciones en tres dimensiones, sino para la simulación, ya sea de estrés mecánico, aerodinámico, incluso del mismo proceso de fabricación a través de las máquinas herramientas.

Estos modelos tridimensionales permiten tener objetos que son transformables a un modelo físico por medio de la impresión 3D, esto es, la fabricación de una pieza física en tres dimensiones directamente de una descripción numérica (típicamente, un modelo CAD) mediante un proceso rápido, totalmente automatizado y flexible sin ninguna herramienta. En otras palabras, de acuerdo con la ASTM (American Society for Testing and Materials, o Sociedad Estadounidense para Pruebas y Materiales), es la fabricación de objetos mediante la deposición de un material utilizando un cabezal de impresión, boquilla u otra tecnología de impresión.

Figura 11. Software de modelado.

El término se utiliza normalmente haciendo referencia a la impresión aditiva, que es el tipo de impresión por boquilla más común, pero la impresión 3D comprende otros métodos:

TECNOLOGÍAS DE IMPRESIÓN 3D	**Extrusión**	Modelado por deposición fundida (FDM).
		Termoplásticos (PLA, ABS), metales eutécticos, materiales comestibles
	Granulado	Proyección aglutinante (polvo)
		Sinterizado por laser (SLS)
		Sinterizado selectivo por calor
		Fusión por haz de electrones
	Laminado (LOM)	Laminado de capas, papel, papel de aluminio, capa de plástico
	Fotoquímicos	Fotopolimerización por luz ultravioleta (SGC)
		Estereolitografía (SLA)

Tabla 1. Software de modelado.

Los tipos más comunes de impresión 3D comprenden tres: el modelado por extrusión (FDM), la estereolitografía (SLA), y el sinterizado por láser (SLS). La impresión FDM (la imagen anterior ilustra este tipo) consiste en depositar polímero fundido sobre una base plana, capa por capa, usa material que inicialmente se encuentra en estado sólido almacenado en rollos, se funde y es expulsado por una boquilla en hilos muy delgados que se solidifican nuevamente al disminuir la temperatura, tomando la forma deseada del modelo. Se trata de la técnica más común y utiliza los típicos termoplásticos, como PLA, ABS, PET, TPU, y otros materiales como cerámicos, metales y biomateriales.

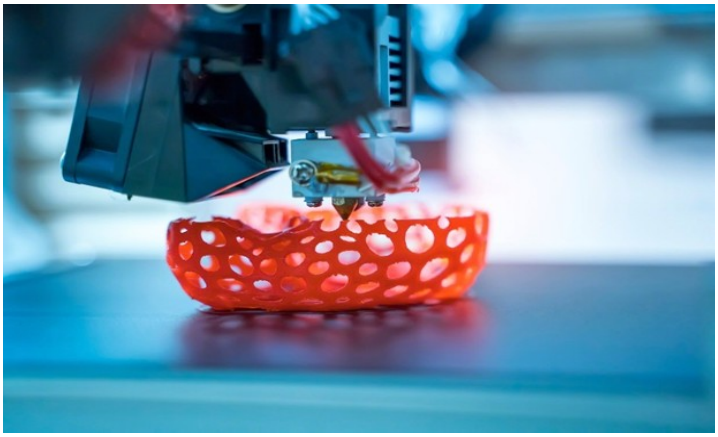


Figura 12. Proceso de Impresión 3D.

Como se mencionó anteriormente, el proceso de fabricación de 3D comienza con un modelo CAD, ya con un modelo terminado y listo para pasar a su creación en físico, se procede a un formato estándar de prototipado rápido, en otras palabras, se convierte en un archivo que pueda ser fácil de interpretar para su procesamiento, por cual se exportan a un archivo con extensión .STL, este formato define las superficies del modelo a uno tridimensional mediante una representación triangularizada (cada triángulo es perpendicular a la superficie) más fácil de interpretar, y por ende, procesar.

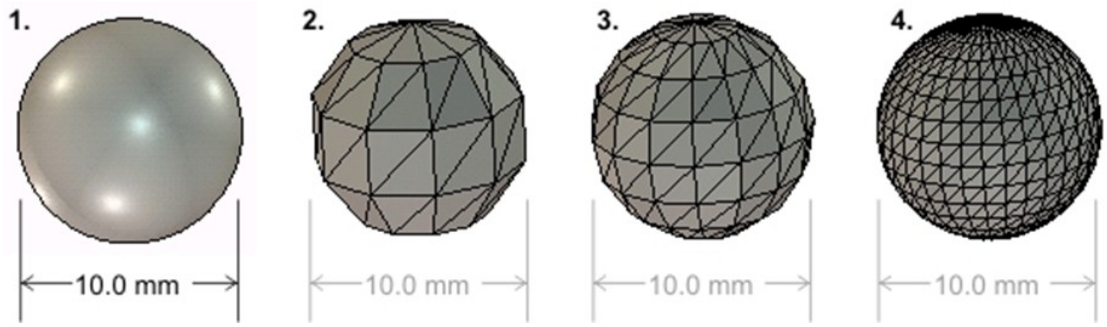


Figura 13. Dimensionado del objeto de impresión.

Posteriormente, el archivo se transforma por medio de un programa a código G, un tipo de código que, por medio de instrucciones de momento, describe los pasos que debe seguir la impresora para crear el modelo, desde las coordenadas, distancias en las tres dimensiones, hasta la velocidad del movimiento y del extrusor de material. Estos programas especiales se conocen como “slicer” (cortadores, esto debido a que dividen el modelo a cada capa correspondiente de acuerdo a la configuración, entre los más conocidos, se encuentra Cura, Netfabb, PrusaSlicer, Simplify3D, MakerBot Print, Creality Slicer, etc.).

Figura 14. Reconstrucción para la impresión.

Al concluir con el proceso de impresión, se requiere un post-procesado que es muy sencillo de realizar ya que necesita de herramientas muy simples, que incluso pueden sustituirse fácilmente en caso de no contar con estos. El material se retira manualmente usando herramientas como espátulas, y se procede a retirar imperfecciones mínimas con pinzas o lija, al menos para impresión tipo FDM.

Figuras 15 y 16. Post procesado de la pieza impresa.

2.9 Materiales de Impresión 3D

Las impresoras 3D usan una amplia gama de materiales, algunos con propiedades físicas excepcionales incluso para uso industrial, o características especiales para un uso específico. Dejando de lado las impresoras de nivel industrial, la mayoría de usuarios de estas máquinas utilizan ciertos materiales que se han extendido en todo el mundo como los más comunes, algunos con más uso que otro dependiendo que tan versátil sean para su uso en general, los materiales más comunes se describen brevemente a continuación:

- PLA – ácido poli láctico: Es un polímero constituido por moléculas de ácido láctico y es actualmente el polímero biodegradable más popular en la industria, siendo usado por todos los aficionados y profesionales para toda clase de tareas, debido a ser el más económico en general.

- PETG: Tereftalato de Polietileno Glicol (Se le agrega Glicol para mejorar su uso en las impresoras) es uno de los materiales más usados para las botellas y otro tipo de envases. Su principal propiedad es su capacidad de cristalización, generando piezas transparentes con efectos sorprendentes. Es muy fuerte y resistente a los impactos. Su densidad cristalina es de 1,45 g/cm3, y se destaca porque en general presenta una buena relación de calidad-precio entre sus propiedades mecánicas y disponibilidad de mercado,

además de no presentar verdaderas dificultades como absorción de humedad o emisión de gases y partículas durante la impresión.

- ABS–Acrilonitrilo butadieno estireno: Es un termoplástico amorfo extraído del petróleo. Presenta buenas propiedades mecánicas que lo hacen adecuado para automoción y aplicaciones industriales, y también para usos domésticos. El costo de producción de estos Polímeros está estrechamente ligado al precio del crudo y no es ecológico debido a la emisión de gases de efecto invernadero durante su producción

- Nylon: Es uno de los materiales más complejos para la impresión 3D. Su principal problema es la falta de adhesión de la pieza a la bandeja, que causa muchos fallos además de un “warping” muy difícil de controlar (warping es la deformación de la impresión al levantarse las esquinas despegándose de la plataforma), pero que compensa con todas sus propiedades. Además, suele coger fácilmente húmeda y debe secarse con anticipación para su uso.

2.10 Servos

Un servomotor es un dispositivo electromecánico que combina un motor eléctrico con un sistema de control y retroalimentación, permitiendo un control preciso de la posición, velocidad y aceleración de su eje. Se utiliza ampliamente en aplicaciones donde se requiere un movimiento exacto y repetible, como en la robótica, maquinaria CNC y sistemas de automatización industrial.

Un servo o servomecanismo es un sistema que funciona según el principio de retroalimentación negativa para inducir una acción que haga que la salida se esclavice a la entrada.

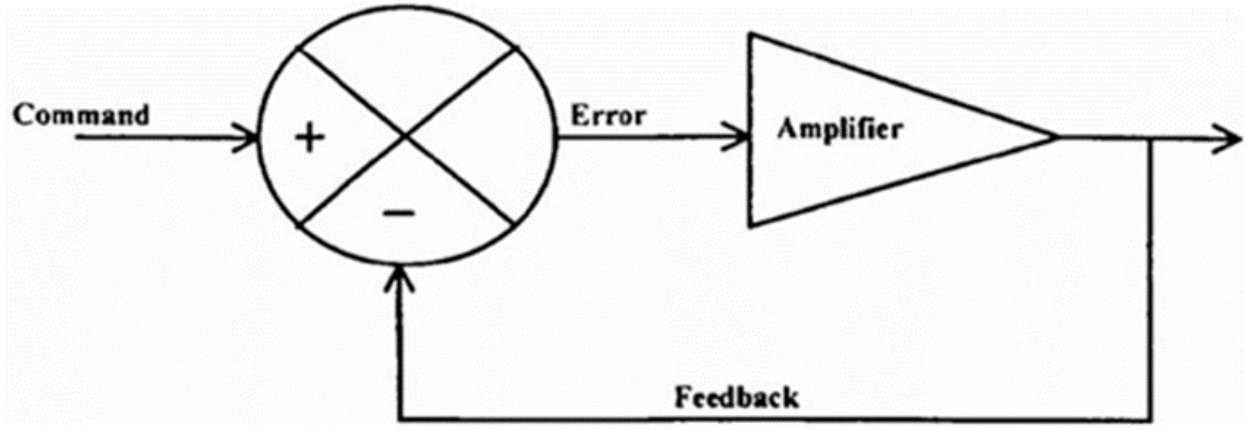


Figura 17. Diagrama esquemático de funcionamiento de un servo.

MG995 (torque alto)

El MG995 es un servo motor de alta calidad producido por TowerPro; un fabricante chino de electrónica y componentes para robótica. Es uno de los servos más populares y ampliamente utilizados en la comunidad de robótica, debido a su gran capacidad de carga, alta velocidad, precisión y durabilidad.

El 48MG995 es un servo digital, lo que significa que se controla mediante señales de pulso digital; su rango de rotación es de 180 grados, con una resolución de 1 grado, lo que lo hace adecuado para una amplia variedad de aplicaciones, desde robots de juguete hasta brazos robóticos industriales.

Dicho servo tiene un par de torsión, máximo de 10 kg/cm a 4,8 V, lo que lo hace adecuado para levantar cargas pesadas y para aplicaciones que requieren un alto nivel de precisión y control de movimiento.

Asimismo, tiene una velocidad de rotación de 0,17 segundos/60 grados a 4,8 V, lo que significa que puede realizar movimientos rápidos y precisos. El MG995 está construido con materiales de alta calidad y tiene una carcasa de plástico resistente que lo hace duradero y resistente a los impactos. Además, cuenta con un sistema de engranajes metálicos que garantiza una

mayor vida útil del motor y una mayor precisión en los movimientos. En suma, el MG995 es un servo motor de alta calidad, confiable y versátil, ampliamente utilizado en la comunidad de robótica, debido a su capacidad de carga, velocidad, precisión y durabilidad.

Figura 18. Servo Motor MG995.

Características:

- Weight: 55 g • Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC

MOT-110 (micro servo de Steren)

Este micro servomotor es ideal para proyectos de mini robótica y prácticas con servomotores. Su bajo consumo energético permite alimentarlo con la misma fuente que tu circuito de control, como una tarjeta Arduino. El mecanismo de engranajes es metálico, lo que le proporciona una larga vida útil. Es compatible con la mayoría de las tarjetas electrónicas de control y

receptores como Futaba o JR gracias a su conector tipo "S".

Este servomotor es una excelente opción para proyectos que requieren precisión y fiabilidad en el control del movimiento mecánico.

Figura 19. Servo motor MOT-110.

Características:

- Torque de 1,8 kgf*cm @ 4,8 Vcc
- Torque de 2,2 kgf*cm @ 6 Vcc
- Rotación de 180°
- Temperatura de operación: -30° a 60°
- Figura 19. Descripción de la imagen 19.
- Velocidad de operación: < 0,1 s
- Conector tipo S compatible con receptores Futaba y JR
- Engranaje metálico de larga vida
- Alimentación: 3,5 - 6 Vcc 40 mA

Características principales de los servomotores:

- Control de posición y velocidad: Gracias a su sistema de retroalimentación, el servomotor puede ajustar su movimiento en tiempo real para alcanzar y mantener una posición o velocidad deseada.

- Sistema de retroalimentación: Utiliza sensores como encoders o potenciómetros para medir la posición actual del eje y compararla con la posición objetivo, ajustando el movimiento según sea necesario.

- Precisión y eficiencia: Diseñados para ofrecer alta precisión en el control del movimiento, los servomotores son ideales para tareas que requieren movimientos exactos y rápidos.

Componentes de un servomotor:

- Motor eléctrico: Puede ser de corriente continua (DC) o alterna (AC), y es el encargado de generar el movimiento (DTC motor).

- Sistema de control: Circuito electrónico que recibe señales de entrada y ajusta el funcionamiento del motor para alcanzar la posición o velocidad deseada. (Encoder?)

- Sensor de retroalimentación: Dispositivo que mide la posición actual del eje del motor, proporcionando información al sistema de control para realizar ajustes precisos.

- Engranajes reductores: Utilizados para aumentar el torque y reducir la velocidad del motor, permitiendo un control más preciso del movimiento.

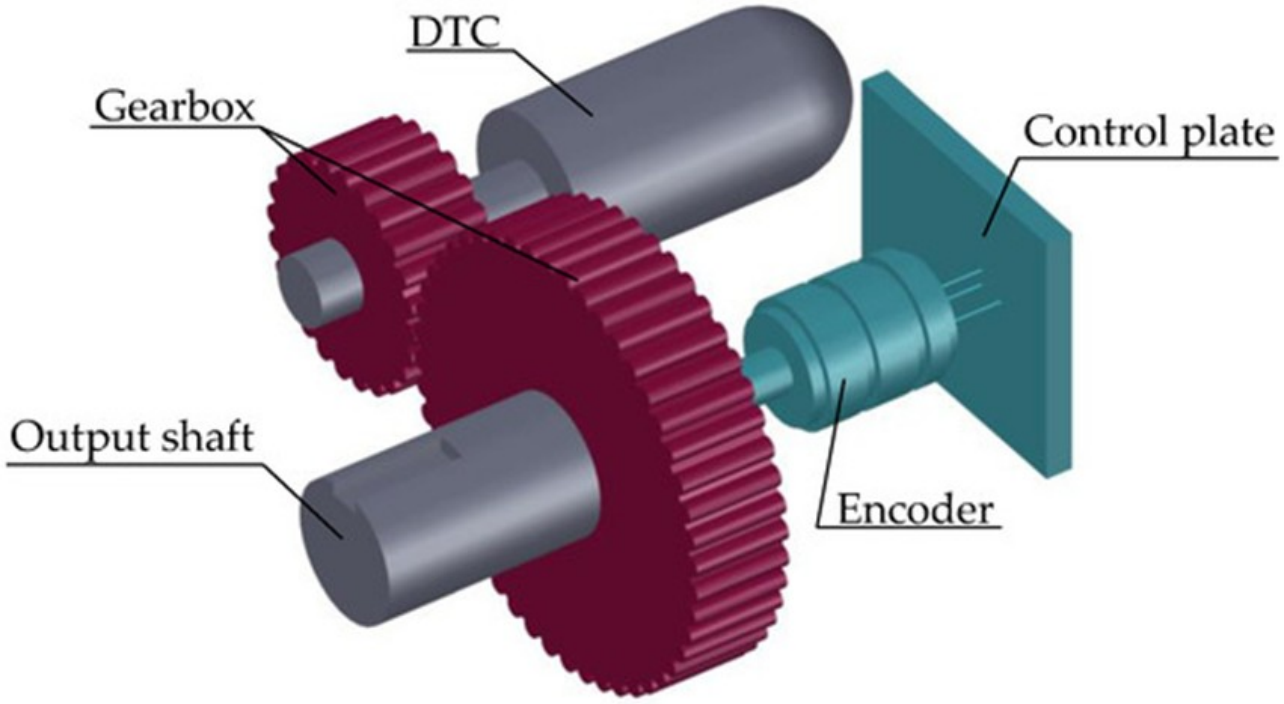


Figura 20. Componentes de un servomotor.

2.11 La modulación por ancho de pulso y su aplicación en Servomotores

La modulación por ancho de pulso (PWM, por sus siglas en inglés) es una técnica de control ampliamente utilizada en electrónica, que permite modificar el valor promedio de una señal de voltaje de corriente directa constante. En lugar de alterar la amplitud de la señal, lo que se varía es la duración del pulso dentro de un periodo fijo, también conocido como ciclo de trabajo. Esta técnica se basa en la generación de señales cuadradas donde se regula el tiempo durante el cual la señal se mantiene en nivel alto, modulando así su efecto sobre una carga o dispositivo.

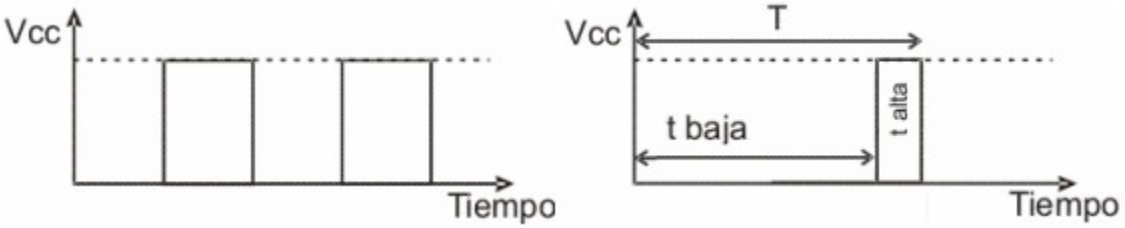


Figura 21 y 22. Modulación de ancho de pulso. Fuente: http://www.forosdeelectronica.com/

En la figura se muestra la forma de aplicar la modulación de ancho de pulso.

Actualmente, numerosos circuitos integrados (CI) incorporan funciones de PWM como parte de su arquitectura. Algunos incluso están diseñados para aplicaciones específicas como el control de fuentes conmutadas o regulación de motores. Entre las empresas que fabrican este tipo de componentes destacan Texas Instruments, Maxim, National Semiconductor, entre otras.

En el ámbito de la robótica, el PWM tiene una aplicación clave en el control de servomotores. El principio consiste en enviar al servo una serie de impulsos de duración variable, manteniendo un periodo de repetición constante. El tiempo que el pulso permanece en nivel alto determina la posición angular del eje del motor. Aunque el periodo puede variar ligeramente, típicamente se utiliza un valor de alrededor de 20 milisegundos.

Cada servo interpreta el ancho del pulso como una instrucción para posicionarse en un ángulo específico.

Por lo general:

- Pulsos de 1 ms colocan el servo en el extremo izquierdo (0°),
- Pulsos de 2 ms en el extremo derecho (180°),
- Pulsos de 1.5 ms indican la posición central (90°).

Algunos servos permiten ángulos superiores a 180°, aceptando pulsos fuera del rango típico, aunque esto depende del diseño interno del servo y sus limitaciones mecánicas. Si el pulso excede los límites aceptables, el servo puede producir un zumbido, indicando que se está forzando su mecanismo interno.

El intervalo entre pulsos no es estrictamente crítico, pero si es

demasiado corto puede interferir con el temporizador interno del servo, provocando vibraciones o comportamientos erráticos. Por el contrario, si el intervalo es demasiado largo, el servo puede entrar en un estado de reposo, lo que resulta en pérdida de fuerza y precisión. Para mantener un servo en una posición estable, es necesario enviar de forma continua el mismo pulso correspondiente a la posición deseada. Esto permite al sistema resistir fuerzas externas y conservar su postura.

Una posible desventaja del uso de PWM es la generación de interferencias electromagnéticas debidas a las rápidas conmutaciones de señal, las cuales pueden afectar otros dispositivos cercanos. Este efecto puede reducirse mediante técnicas como el filtrado de la fuente de alimentación y la colocación estratégica de los componentes electrónicos dentro del circuito.

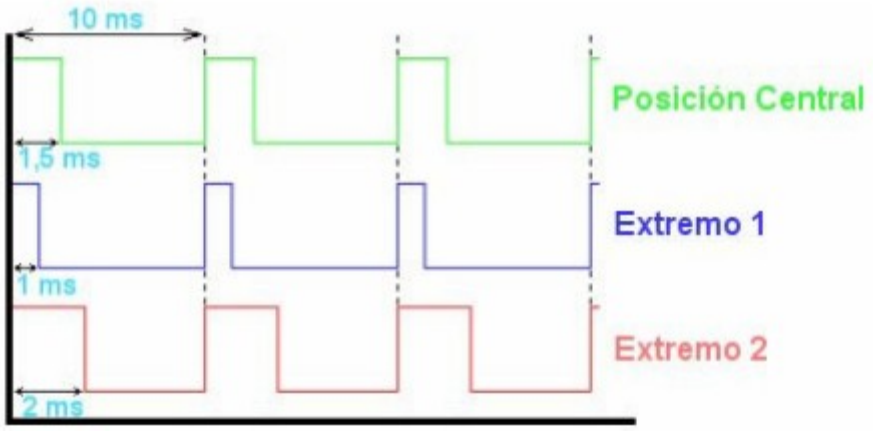


Figura 23. Ejemplificación del PWM. Fuente: http://cfievalladolid2.net

2.12 Lab View

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un entorno de programación visual ampliamente utilizado en aplicaciones de adquisición de datos, control de instrumentos y automatización de procesos. Este software se distingue por permitir el diseño gráfico de sistemas a través

de bibliotecas intuitivas y versátiles que facilitan todas las etapas del análisis de mediciones: desde la adquisición, pasando por el procesamiento, hasta la visualización de datos.

El lenguaje de programación utilizado en LabVIEW se denomina Lenguaje G, y su enfoque gráfico lo hace accesible incluso para usuarios sin experiencia previa en programación tradicional. Originalmente desarrollado por la empresa National Instruments en 1976 para plataformas Macintosh, LabVIEW fue lanzado comercialmente en 1986 y, desde entonces, ha sido adaptado para funcionar en diversos sistemas operativos, incluyendo Windows, UNIX, Linux y versiones modernas de macOS.

Los programas creados en este entorno reciben el nombre de Instrumentos Virtuales (VIs). Esta denominación hace referencia al hecho de que estos programas están diseñados para simular el funcionamiento de instrumentos físicos, como osciloscopios, multímetros y generadores de señales. La interfaz gráfica de usuario emula el panel frontal de estos dispositivos, lo que facilita la interacción del usuario con las variables de entrada y salida del sistema.

Uno de los principales lemas de LabVIEW es: “La potencia está en el software”, reflejando su enfoque en reducir los tiempos de desarrollo de aplicaciones complejas y facilitar el acceso a soluciones informáticas incluso para usuarios con poca formación técnica. Aun así, el software está diseñado para integrarse con una amplia gama de hardware, como tarjetas de adquisición de datos, sistemas embebidos (PAC), equipos de visión artificial, entre otros, incluyendo compatibilidad con dispositivos de terceros.

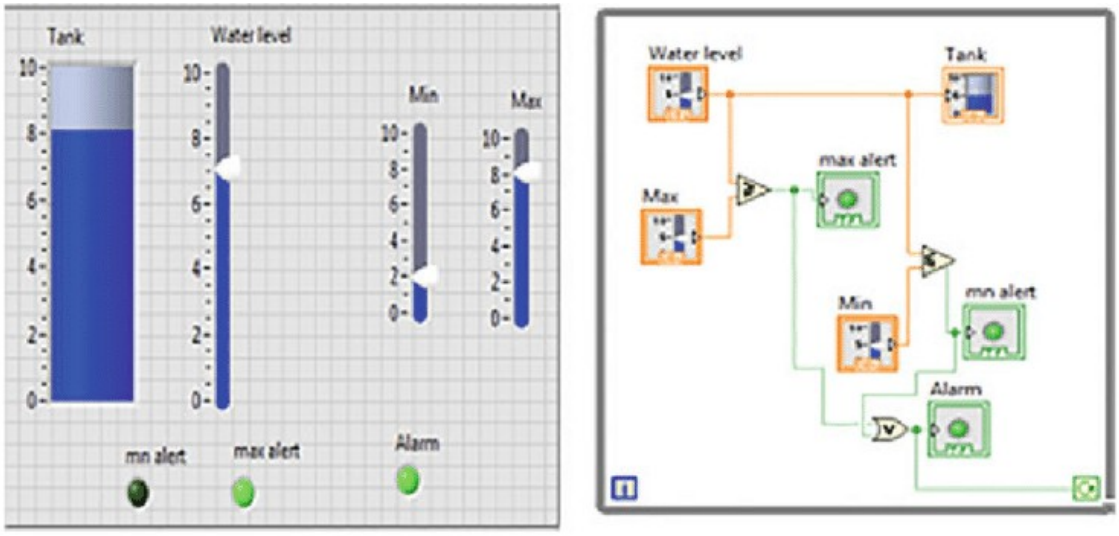
LabVIEW ofrece un conjunto amplio de herramientas destinadas a la adquisición, análisis y almacenamiento de datos, además de utilidades para la depuración y resolución de errores en el código. El desarrollo de un VI implica trabajar con dos ventanas principales:

4. Panel Frontal:

Es la interfaz visual con la que el usuario interactúa. Aquí se colocan controles (como botones, perillas o entradas numéricas) y indicadores (como gráficas, luces LED o displays digitales). Estos elementos permiten tanto ingresar como visualizar datos durante la ejecución del programa.

5. Diagrama de Bloques:

Es la zona donde se construye el programa mediante la conexión gráfica de bloques funcionales, siguiendo una lógica de flujo de datos. Aquí se determinan las relaciones y operaciones que se realizan sobre las señales o variables manejadas por el sistema.



Figuras 24 y 25. Ventana de Front Panel y Block Diagram en Lab View.

Esta estructura visual permite que la programación en LabVIEW sea considerada modular, dado que cada bloque representa una función específica que puede ser reutilizada o modificada de forma independiente. La lógica del programa depende del orden y conexión de estos bloques, similar al funcionamiento de un diagrama de flujo.

2.13 Arduino

Para manejar un robot de cualquier tipo es necesario el uso de un

controlador, para proyectos relativamente pequeños se puede optar por un microcontrolador, que es una plataforma de hardware libre. Dispone de un circuito integrado a través del cual se pueden grabar instrucciones. A su vez, estas instrucciones se escriben utilizando un lenguaje de programación que permite al usuario establecer programas que interactúan con los circuitos electrónicos.

Arduino es una placa electrónica barata, disponible en el mercado, con un microcontrolador y algunas capacidades de E/S. Existe en varias versiones, que comparten el mismo y sencillo lenguaje de programación, debido a su disponibilidad es una excelente opción para que sea el cerebro de un brazo robot didáctico. Las funciones de Arduino, como ocurre con la mayoría de las placas de microcontroladores, se pueden resumir en 3 factores:

1. Dispone de una interfaz de entrada. Ésta puede estar directamente vinculada a los periféricos, o conectarse a ellos a través de puertos.
2. El propósito de la interfaz de entrada es transferir la información al microcontrolador. El microcontrolador es la pieza que se encarga de procesar estos datos. Además, varía en función de las necesidades del proyecto en el que se quiera utilizar la placa, y existe una gran variedad de fabricantes y versiones disponibles.
3. También dispone de una interfaz de salida. Ésta se encarga de llevar la información procesada a los periféricos autorizados para hacer el uso final de esos datos. En algunos casos, puede ser otra placa en la que se centraliza la información y se procesa de forma totalmente renovada, o simplemente, puede ser una pantalla o un altavoz encargado de mostrar la versión final de los datos.

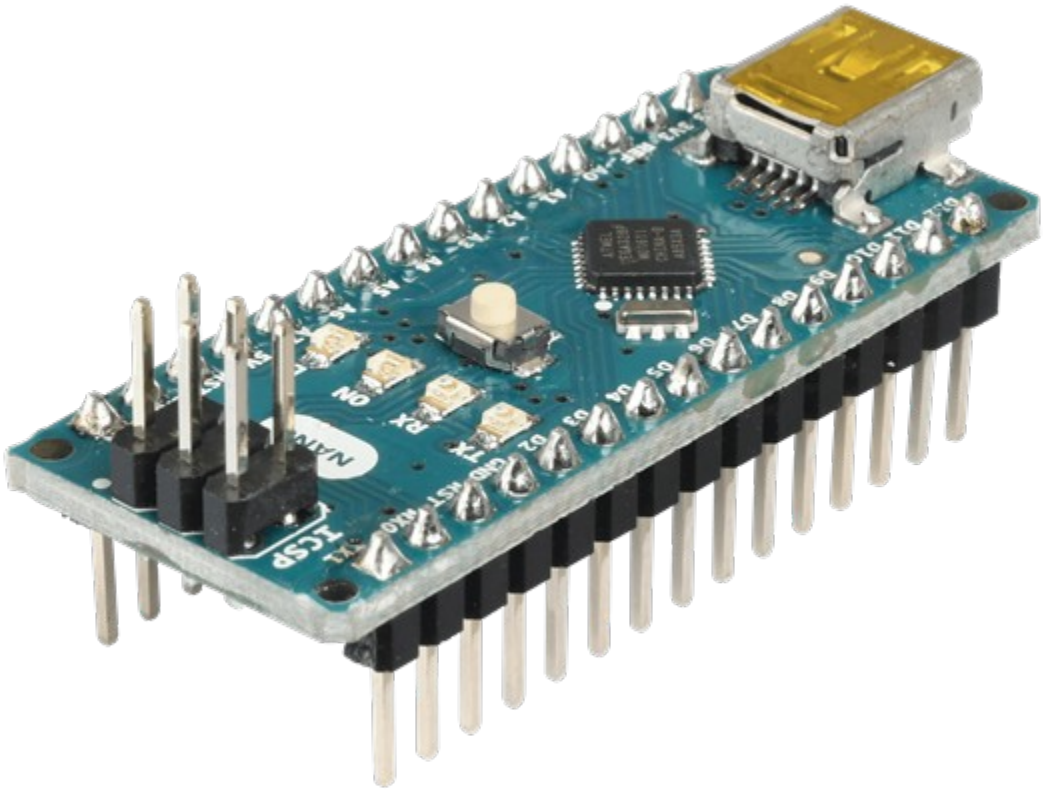


Figura 26. Placa de Arduino NANO.

3. DESARROLLO.

Este proyecto sigue un enfoque tecnológico-aplicado, ya que busca el desarrollo e implementación de un brazo robótico funcional con fines educativos. Se empleará una metodología de diseño iterativo donde combinaremos elementos de ingeniería mecánica, electrónica y programación para lograr un prototipo funcional que permita cumplir con los objetivos del proyecto.

Como punto de partida, se recibió del asesor académico un archivo en formato stl; correspondiente a un modelo preliminar para el brazo robótico didáctico. Para trabajar el modelo utilizamos Fusión 360, es un software CAD

de modelado 3D basado en la nube desarrollado por Autodesk. Es especialmente adecuado para la creación rápida de prototipos y flujos de trabajo integrados desde el diseño hasta la fabricación. Se proporciona gratuitamente a los estudiantes proporcionando el correo institucional, con renovación anual.

Figura 27. Programa de Fusion 360.

El diseño inicial del brazo se conservó en los archivos de clase STL, lo cual se puede observar en el aspecto descompuesto en triángulos de las piezas que no pudieron exportarse correctamente a archivo .STEP debido a su geometría.

Figura 28. Modelo 3D del brazo robótico.

3.1 Impresión del modelo 3d inicial y postprocesado de piezas.

El archivo STL fue abierto con el programa “slicer”: Creality Print, para configurarlo se hace uso de diferentes parámetros para determinar tanto la duración y material usado en la impresión, los parámetros también afectan qué tan resistente es la impresión. Los parámetros vitales a modificar serían los siguientes:

- Densidad de relleno:

Se refiere a la estructura interna de una pieza impresa en 3D que rellena el espacio entre las capas exteriores. El relleno sirve como estructura de soporte interno para los objetos impresos en 3D, garantizando su integridad estructural. Mientras que algunos procesos de fabricación requieren que las piezas sean completamente sólidas o huecas, la impresión 3D permite la creación de intrincados patrones de relleno que optimizan la resistencia, el peso y el uso de materiales. Las densidades de relleno más altas dan como resultado objetos más resistentes, ya que hay más material que soporta las carcasas exteriores.

El software Slicer permite a los usuarios ajustar el porcentaje de densidad de relleno, que va del 0% (hueco) al 100% (sólido). Las piezas huecas requieren menos tiempo y material, pero pueden sacrificar resistencia y durabilidad. La selección de un porcentaje de relleno adecuado depende del uso previsto de la pieza impresa en 3D. Estas son algunas recomendaciones generales:

- 0-20% Piezas no funcionales: Para piezas que no necesitan soportar fuerzas significativas, como modelos de exhibición o prototipos de presentación, basta con un porcentaje de relleno bajo. El relleno cero puede ser una opción si no hay grandes superficies planas en la parte superior de la impresión.
- 20-40% Piezas de uso ligero: Las piezas que van a soportar cierta fuerza, pero no cargas pesadas, pueden beneficiarse de un porcentaje

de relleno medio. Esto proporciona una resistencia adecuada al tiempo que reduce el uso de material y los costes.

- 40-100% Piezas de uso intensivo: Las piezas que requieren una gran resistencia y durabilidad para soportar fuerzas significativas deben tener un porcentaje de relleno alto. Sin embargo, aumentar el porcentaje de relleno por encima del 60% puede incrementar significativamente el tiempo de impresión y el consumo de material.

Dado que primero imprimimos el primer modelo con los archivos iniciales, utilizamos el relleno de 25%, el software permite utilizar diferentes patrones de relleno, se utilizó el predeterminado dado que tiene una buena relación de velocidad e integridad estructural.

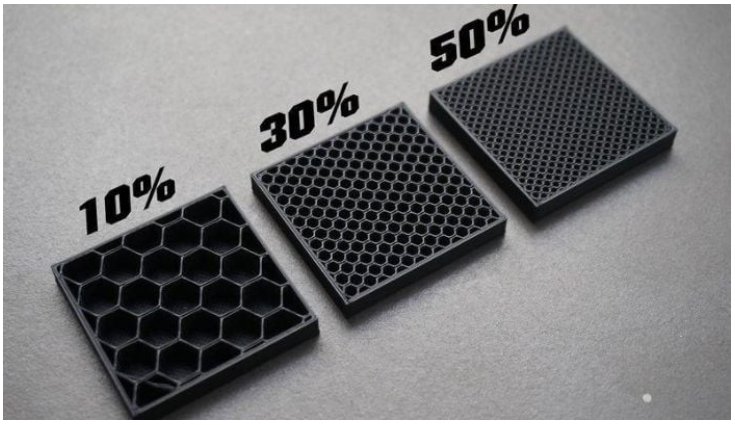


Figura 29. Apariencia interna promedio de una pieza conforme a su porcentaje de relleno.

- Número de capas superficiales:

Esto se refiere al número total de capas de las paredes internas y externas combinadas. El muro encierra una parte de relleno con menor densidad y resistencia, que ocupa una gran parte del volumen total del modelo tridimensional. Por lo tanto, cuantas más capas de pared haya, más gruesa será la pared, mayor será la resistencia de la superficie y menor será la proporción del patrón de relleno. Esto da como resultado una mayor resistencia estructural general del modelo. Sin embargo, tener demasiadas capas de pared no siempre es mejor, ya que puede hacer que el modelo

impreso se vuelva más sólido, aumentando así el tiempo de impresión. Normalmente, basta con establecer el número de capas en 2-3.

Para el prototipo se usó 2 capas superficiales:

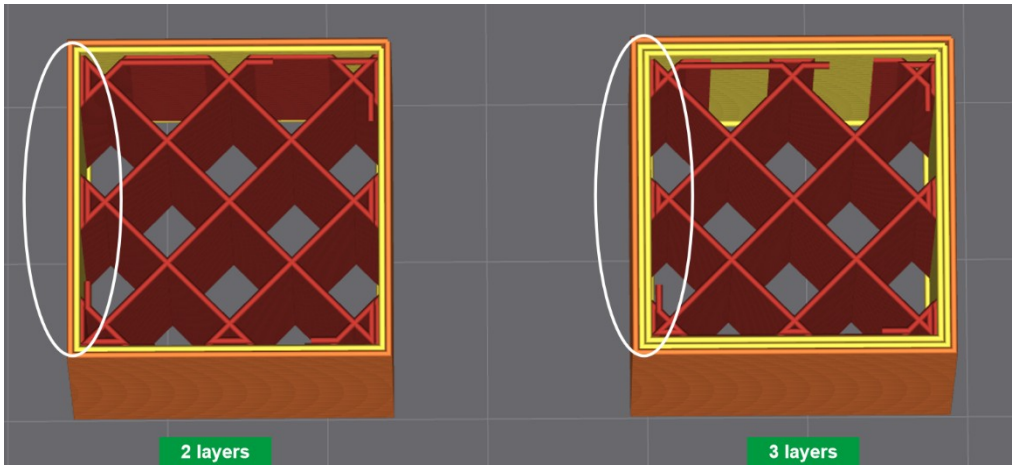


Figura 30. Diferencia de solidez de una impresión 3D según el número de capas superficiales.

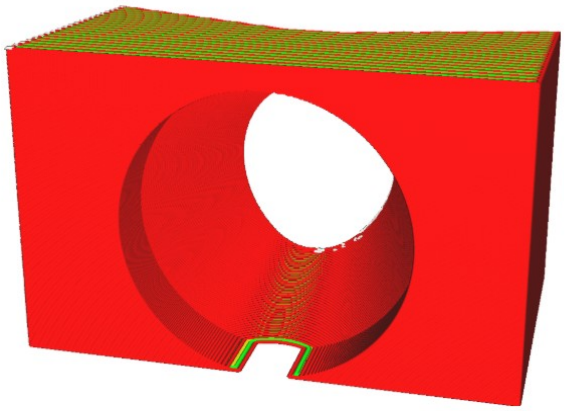
- Altura de capa:

La altura de cada capa en el proceso de apilamiento de plástico capa por capa representada por la impresora 3D representa la altura de la capa. La altura de la capa es el grosor de estas capas en milímetros. Es el factor más importante que afecta a la calidad visual y al tiempo de impresión de la impresión final. El siguiente es un resumen de algunos de los impactos de los cambios en la altura de la capa (más delgada, más gruesa):

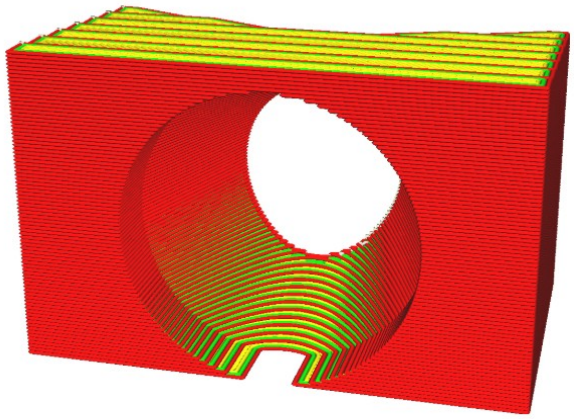
- Las alturas de capa más delgadas mejoran la calidad visual de sus impresiones. Como la altura de la capa es más delgada, se reduce el efecto de escalón entre capas. Además, las arrugas entre las capas se harán más pequeñas y la superficie general parecerá más lisa.

- Las alturas de capa más delgadas permiten que la impresora produzca más detalles en la parte superior e inferior de sus impresiones.

- Una altura de capa más gruesa hará que el objetivo de impresión sea algo más fuerte. Habrá menos límites entre las capas, y estos límites suelen ser puntos débiles. Las capas más gruesas no producirán un cizallamiento excesivo.
- Las alturas de capa más gruesas reducen los tiempos de impresión porque la boquilla no necesita moverse tanto horizontalmente.



Altura de capa de 0,1 mm



Altura de capa de 0,3 mm

Figuras 31 y 32 . Nivel de detalle de la impresión respecto a altura de capa.

Para mantener la calidad para mantener el detalle necesario en ranuras, guías, agujeros, etc., se mantendrá en todo momento una altura de capa de 0.2 mm. La velocidad si bien puede modificarse, depende de cada modelo de impresora 3D, para este proyecto se utilizó la impresora de Creality Ender 3 V3, compatible con el software.

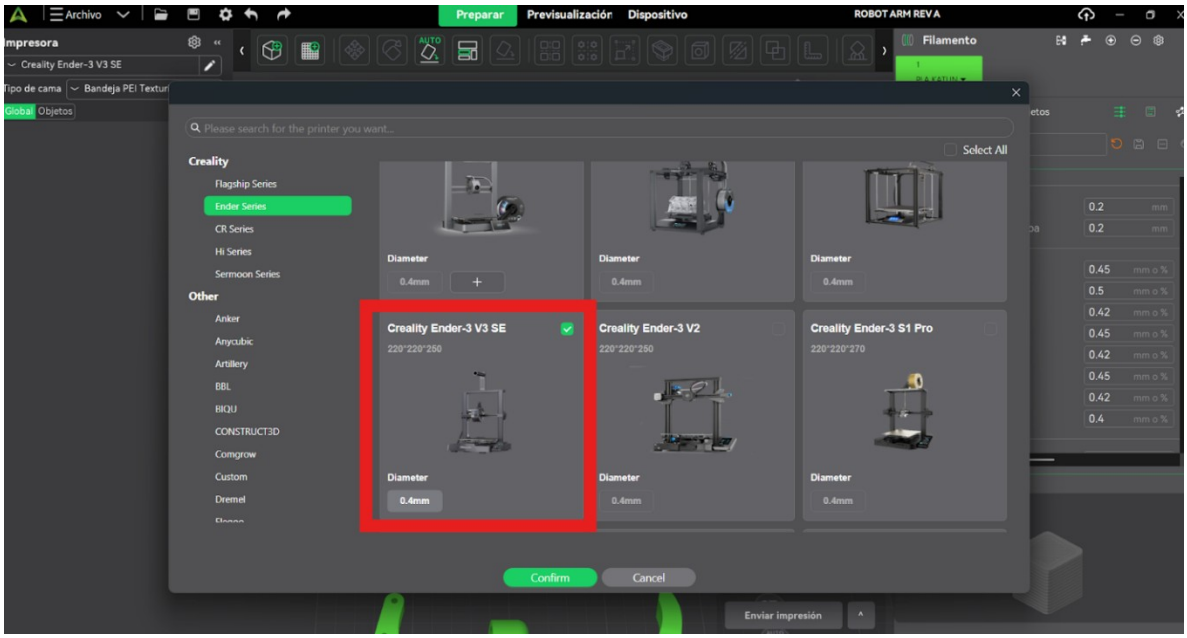


Figura 33. Selección de impresora en Creality

Ahora bien, para imprimir todas las piezas del robot se tuvo que dividir la impresión de todas las piezas entre varias bandejas de impresión.

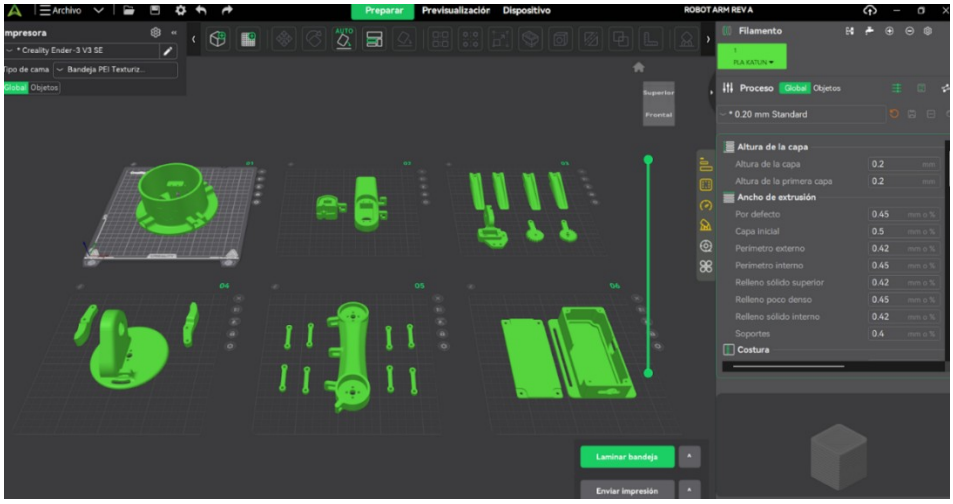
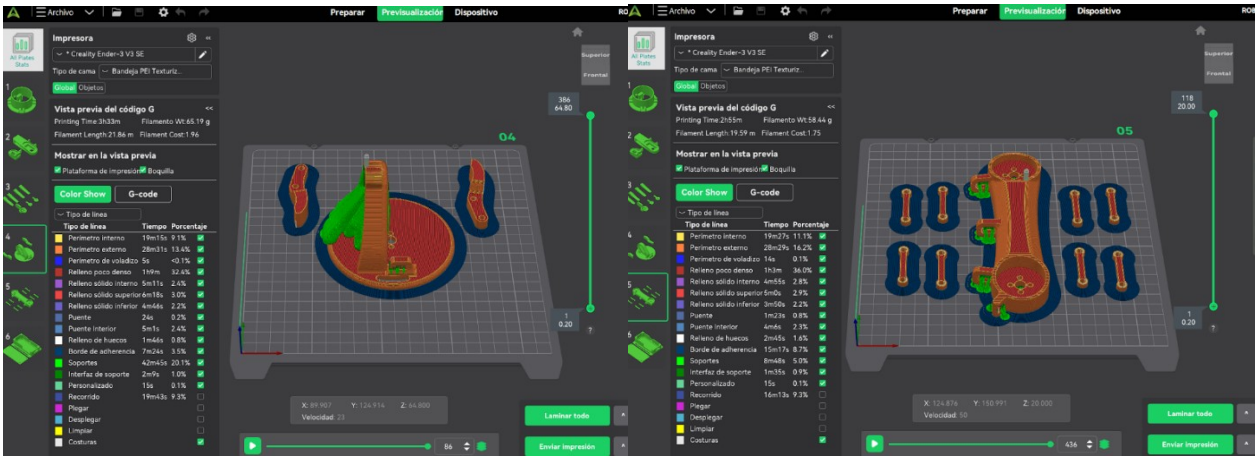
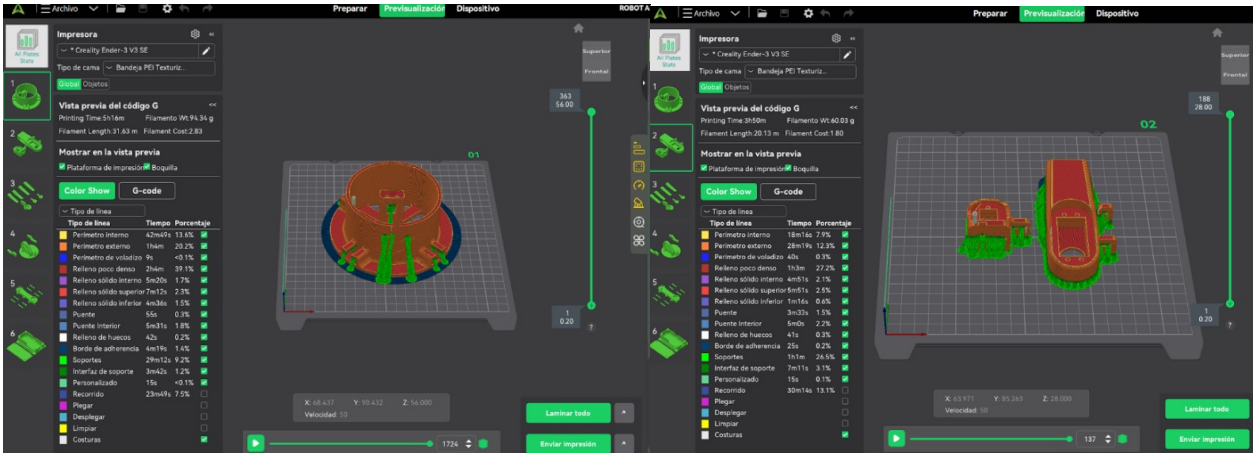


Figura 34. Organizado de piezas por bandejas para un equilibrio de tiempo de impresión.

Aquí también se pueden ver las especificaciones de impresión que se utilizaron para cada una de las bandejas.





Figuras 35, 36, 37, 38. Especificaciones de impresión por bandeja.

El modelo fue impreso usando plástico PLA, que era el disponible para fabricar el prototipo.

Después de la impresión, se retiraron los soportes que mantienen los detalles en voladizo de la pieza durante la impresión, de modo que no quede ningún material extra en la pieza que pueda ser obstáculo para el cableado o montaje de los motores.

3.2 Modelado y análisis teórico.

A la par de la impresión del modelo, utilizamos el programa de fusión para analizar la morfología del robot, esto nos permitirá describir la ecuación de movimiento del robot, el comportamiento cinemático y estructural del brazo robótico mediante un análisis teórico que permita prever su desempeño, facilitar el diseño del control y justificar las decisiones tomadas en el desarrollo físico.

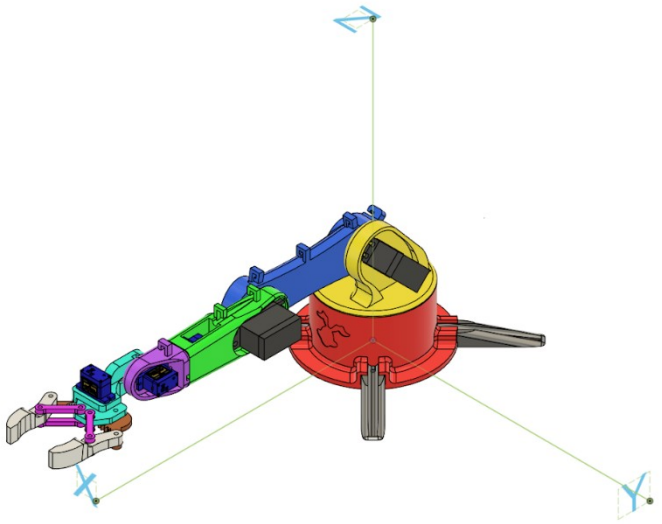


Figura 39. Posición de origen del robot en vista isométrica de Fusión 360.

3.2.1 Ecuación de movimiento del robot

El análisis de la configuración del brazo robótico nos permitió determinar los grados de libertad, el tipo de articulaciones y la estructura general. El modelo está compuesto por cinco articulaciones rotacionales (R), que permiten movimientos en diferentes planos, lo que facilita una manipulación versátil dentro de un espacio tridimensional.

Para desarrollar la ecuación de movimiento de un robot, primero es esencial entender la morfología del robot y los componentes que lo constituyen. En este caso, estamos considerando un brazo robótico de 5 grados de libertad (DOF), que es comúnmente utilizado en aplicaciones educativas. Este tipo de robot generalmente está compuesto por una serie de eslabones conectados por articulaciones rotacionales.

1. Articulaciones y Eslabones:

- El robot tiene cinco articulaciones rotacionales, lo que significa que cada articulación permite un movimiento de rotación alrededor de un eje fijo.

- Los eslabones son las partes rígidas que conectan las articulaciones. En este robot, los eslabones tienen diferentes longitudes, que son cruciales para determinar el alcance y la capacidad de movimiento del robot.

2. Tamaño de los Eslabones:

- El primer eslabón, que conecta la base con la primera articulación, tiene una longitud de aproximadamente 42.625 mm.
- El segundo y tercer eslabón miden lo mismo, 120 mm.
- El cuarto eslabón y más largo, mide 128.1875 mm.

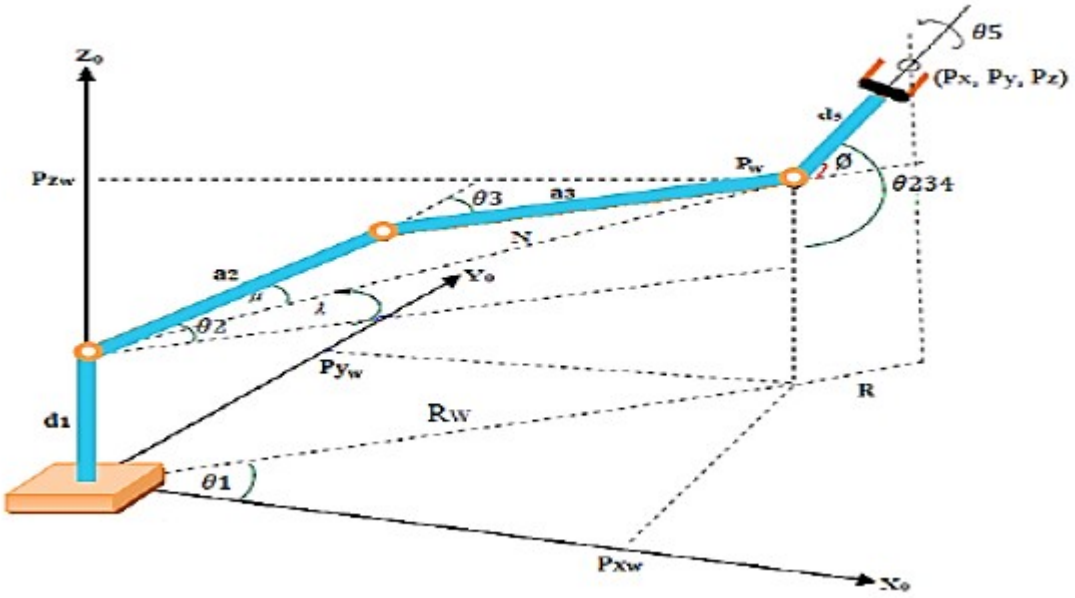


Figura 40. Morfología del robot.

3.2.2 Cinemática Directa

Se desarrolla la cinemática directa del sistema mediante el uso del

método Denavit-Hartenberg (D-H), con el cual se obtiene la posición y orientación del efector final a partir de los ángulos de cada articulación.

Podemos apreciar en esta tabla los parámetros de Denavit-Hartenberg:

Junta i	θ_i (variable)	d_i (mm)	a_i (mm)	α_i (rad)
1	θ_1	$L_0 = 42.625$	0	$\frac{\pi}{2}$
2	θ_2	0	$L_1 = 120$	0
3	θ_3	0	$L_2 = 120$	0
4	θ_4	0	0	$\frac{\pi}{2}$
5	θ_5	$L_3 = 128.1875$	0	0

Tabla 2. Tabla de Denavit-Hartenberg para el brazo robótico de estudio.

La matriz de transformación homogénea se construye para cada eslabón, y luego se multiplica para obtener la posición final del efector respecto al sistema de coordenadas base.

Aquí se muestran las matrices generadas para el cálculo de la cinemática directa

Una vez definidos los parámetros Denavit-Hartenberg y obtenidas las matrices de transformación homogénea correspondientes a cada articulación del robot, el siguiente paso consiste en implementar estos cálculos mediante el uso del software MATLAB. Utilizamos MATLAB para calcular estas ecuaciones de movimiento, esta herramienta permite automatizar la multiplicación de matrices y obtener de manera precisa la posición y orientación del efector final en función de los valores de las variables articulares. A continuación, se presenta el desarrollo del código en MATLAB utilizado para realizar dicho análisis cinemático.

% Definición de las longitudes de los eslabones (mm)

```
L1 = 42.625;
L2 = 120;
L3 = 120;
L4 = 128.1875;

% Ángulos de las articulaciones en grados (puedes modificarlos)
theta1 = deg2rad(0);
theta2 = deg2rad(0);
theta3 = deg2rad(90);
theta4 = deg2rad(-0);
theta5 = deg2rad(0);

% Cálculo de las matrices de transformación homogénea de forma explícita

% Matriz de transformación de la base a la articulación 1 (T0_1)
T0_1 = [
    cos(theta1),  0,  sin(theta1),  0;
    sin(theta1),  0, -cos(theta1),  0;
    0,          1,  0,          L1;
    0,          0,  0,          1
];

% Matriz de transformación de la articulación 1 a la articulación 2 (T1_2)
T1_2 = [
    cos(theta2), -sin(theta2),  0,  L2*cos(theta2);
    sin(theta2),  cos(theta2),  0,  L2*sin(theta2);
    0,          0,          1,  0;
    0,          0,          0,  1
];
```

```
% Matriz de transformación de la articulación 2 a la articulación 3 (T2_3)
T2_3 = [
   cos(theta3), -sin(theta3),  0,  L3*cos(theta3);
   sin(theta3),  cos(theta3),  0,  L3*sin(theta3);
   0,           0,           1,  0;
   0,           0,           0,  1
];

% Matriz de transformación de la articulación 3 a la articulación 4 (T3_4)
T3_4 = [
   cos(theta4),  0,  sin(theta4),  0;
   0,           1,  0,           0;
   -sin(theta4), 0,  cos(theta4),  0;
   0,           0,  0,           1
];

% Matriz de transformación de la articulación 4 a la articulación 5 (T4_5)
T4_5 = [
   cos(theta5), -sin(theta5),  0,  0;
   sin(theta5),  cos(theta5),  0,  0;
   0,           0,           1,  L4;
   0,           0,           0,  1
];

% Multiplicación de matrices para obtener la transformación total (T0_5)
T0_5 = T0_1 * T1_2 * T2_3 * T3_4 * T4_5;

% Extraer la posición del efector final
px = T0_5(1,4);
py = T0_5(2,4);
pz = T0_5(3,4);
```

```
% Extraer la orientación (matriz de rotación)
R = T0_5(1:3, 1:3);

% Mostrar resultados
fprintf('Posición del efector final:\n');
fprintf('px = %.2f mm\n', px);
fprintf('py = %.2f mm\n', py);
fprintf('pz = %.2f mm\n', pz);

fprintf('\nMatriz de rotación del efector final:\n');
disp(R);
```

3.2.3 Cinemática Inversa

Ahora que contamos con la matriz de transformación homogénea y los parámetros Denavit-Hartenberg (DH), podemos proceder a la cinemática inversa, que consiste en obtener los valores de las variables articulares a partir de una posición y orientación deseada del efector final (es decir, a partir de la matriz de transformación total).

Una vez obtenida la matriz de transformación homogénea del efector final y conocidos los parámetros de Denavit-Hartenberg del robot, se procede a calcular la cinemática inversa, es decir, determinar los ángulos articulares θ_1, θ_2, θ_3, θ_4 y θ_5 que permiten alcanzar una posición deseada del efector final, definida por sus coordenadas cartesianas (px, py, pz).

Paso 1: Definición de la posición deseada.

Se considera una posición objetivo del efector final dada por:

px = 180 mm

py = 0 mm

$pz = 157$ mm

Paso 2: Suposición inicial

Para simplificar el análisis, se considera que la orientación del efector está alineada con el eje final, y se asume:

$\theta_5 = 0$

Paso 3: Cálculo de θ_1

El primer ángulo articular, que corresponde a la rotación en el plano horizontal, se calcula mediante:

$\theta_1 = \text{atan2}(py, px)$

Paso 4: Cálculo de la posición de la muñeca (wrist center)

La posición del centro de la muñeca se obtiene restando el efecto del último eslabón (L_4) a lo largo de la dirección del efector:

$wx = px - L_4 * \cos(\theta_1) * \cos(\theta_5)$

$wy = py - L_4 * \sin(\theta_1) * \cos(\theta_5)$

$wz = pz - L_4 * \sin(\theta_5)$

Paso 5: Cálculo del ángulo θ_3 (Ley del coseno)

El ángulo del tercer eslabón se obtiene aplicando la ley del coseno sobre el triángulo formado por los eslabones L_2, L_3 y la distancia al centro de la muñeca:

$D = [wx^2 + wy^2 + (wz - L_1)^2 - L_2^2 - L_3^2] / (2 * L_2 * L_3)$

$\theta_3 = \text{atan2}(\sqrt{(1 - D^2)}, D)$

Paso 6: Cálculo del ángulo θ_2

Se usan las siguientes relaciones:

$s_2 = [(wz - L_1)(L_2 + L_3\cos(\theta_3)) - L_3\sin(\theta_3)\sqrt{(wx^2 + wy^2)}] / [wx^2 + wy^2 +$

$(wz - L_1)^2]$

$c_2 = [(wz - L_1)L_3\sin(\theta_3) + (L_2 + L_3\cos(\theta_3))\sqrt{(wx^2 + wy^2)}] / [wx^2 + wy^2 + (wz - L_1)^2]$

$\theta_2 = \text{atan2}(s_2, c_2)$

Paso 7: Cálculo del ángulo θ_4

El ángulo del cuarto eslabón se calcula considerando el desfase angular respecto a los eslabones anteriores:

$\theta_4 = \text{atan2}(-\sin(\theta_5), \cos(\theta_5)) - \theta_2 - \theta$

Dado que no todos los sistemas permiten una solución analítica simple, se puede considerar el uso de métodos numéricos, aproximaciones geométricas o soluciones específicas para trayectorias conocidas. Esta parte es clave para el control del brazo mediante coordenadas espaciales y la realizaremos igualmente con la herramienta de Matlab.

```
% Definición de las longitudes de los eslabones
L1 = 42.625; % mm
L2 = 120; % mm
L3 = 120; % mm
L4 = 128.1875; % mm
% Definición de los parámetros de Denavit-Hartenberg
% [theta, d, a, alpha]
DH_params = [
    0, L1, 0, pi/2;
    0, 0, L2, 0;
    0, 0, L3, 0;
    0, 0, 0, pi/2;
    0, L4, 0, 0
];
```

```
% Posición deseada del efector final
px = 180; % mm
py = 0;   % mm
pz = 157; % mm

% Cálculo de la cinemática inversa
% Asumimos que el ángulo de la muñeca es cero dado que es una articulación meramente
theta5 = 0;
% Cálculo del ángulo theta1
theta1 = atan2(py, px);

% Cálculo de la posición de la muñeca
wx = px - L4 * cos(theta1) * cos(theta5);
wy = py - L4 * sin(theta1) * cos(theta5);
wz = pz - L4 * sin(theta5);

% Cálculo de theta2 y theta3 usando la ley de cosenos
D = (wx^2 + wy^2 + (wz - L1)^2 - L2^2 - L3^2) / (2 * L2 * L3);
theta3 = atan2(sqrt(1 - D^2), D);

% Cálculo de theta2
s2 = ((wz - L1) * (L2 + L3 * cos(theta3)) - L3 * sin(theta3) * sqrt(wx^2 + wy^2)) / (wx^2 + wy^2 + (wz - L1)^2);
c2 = ((wz - L1) * L3 * sin(theta3) + (L2 + L3 * cos(theta3)) * sqrt(wx^2 + wy^2)) / (wx^2 + wy^2 + (wz - L1)^2);
theta2 = atan2(s2, c2);

% Cálculo de theta4
theta4 = atan2(-sin(theta5), cos(theta5)) - theta2 - theta3;
```

```
% Mostrar los resultados
fprintf('Theta1: %.2f degrees\n', rad2deg(theta1));
fprintf('Theta2: %.2f degrees\n', rad2deg(theta2));
fprintf('Theta3: %.2f degrees\n', rad2deg(theta3));
fprintf('Theta4: %.2f degrees\n', rad2deg(theta4));
fprintf('Theta5: %.2f degrees\n', rad2deg(theta5));
```

3.2.4 Análisis de los Límites Mecánicos

Se analizan las restricciones físicas del brazo, como los ángulos máximos permitidos en cada articulación, la longitud de los eslabones y la estructura de montaje.

También se considera el alcance del brazo, la zona muerta (zonas inaccesibles) y posibles interferencias entre piezas durante el movimiento.

Para garantizar la integridad del movimiento del brazo robótico y prevenir posibles colisiones o esfuerzos excesivos en las articulaciones, se implementó una verificación de seguridad en el software de control. Esta validación asegura que los ángulos q1 a q4 se mantengan dentro del rango físico permitido de −90° a 90°. Si alguno de estos valores excede dicho rango, el sistema emite un mensaje de advertencia y detiene la ejecución para evitar daños en la estructura o fallos en la operación.

3.2.5 Simulación Preliminar

Al realizar el diseño en Fusion 360 se quiso realizar una simulación, a pesar de que el software cuenta con su propia extensión de simulación, es un programa relativamente nuevo comparado con los demás softwares CAD, por lo cual para esta sección se optó por utilizar el programa Solidworks para realizar un análisis estático para las piezas, optando por el uso de este software al ser el más conocido entre los ingenieros en general en la región. Dado que las impresiones 3D no son objetos completamente sólidos y solidworks solo asume sólidos rellenos en su totalidad, se tuvo que hacer un

ajuste en las propiedades del material creado para el análisis, el cual se configuró para que la masa de la pieza coincidiera con el estimado de las impresiones en el programa slicer (las versiones finales, el prototipo se tomó como innecesario comprobar).

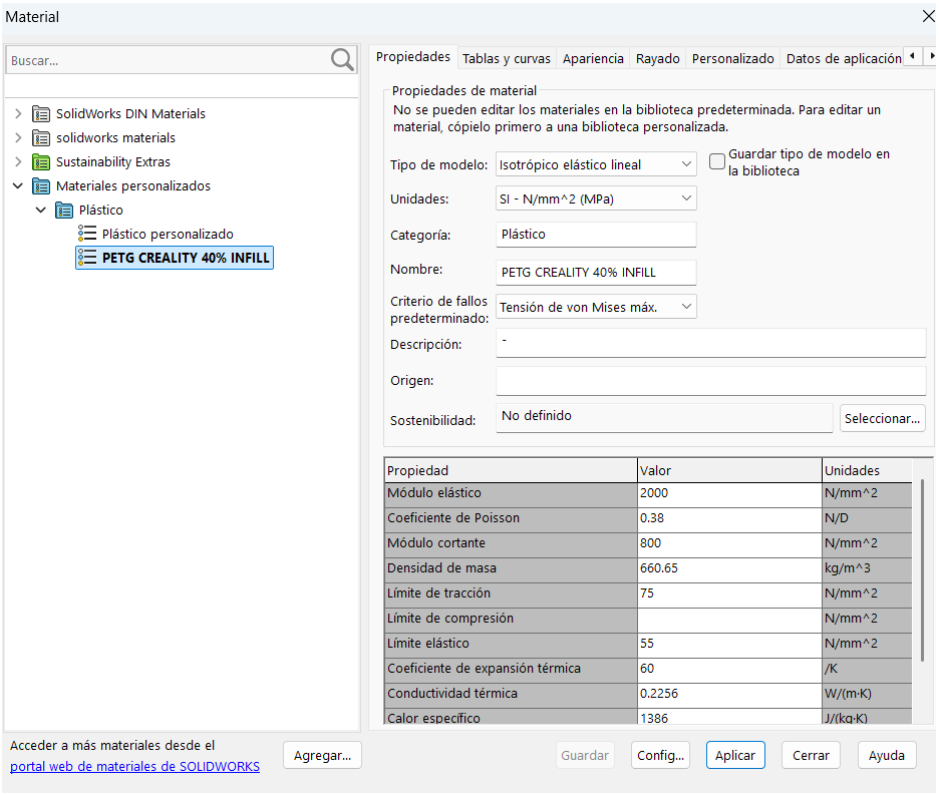

Material
Buscar...
SolidWorks DIN Materials
solidworks materials
Sustainability Extras
Materiales personalizados
Plástico
Plástico personalizado
PETG CREALITY 40% INFILL
Propiedades
Tablas y curvas
Apariencia
Rayado
Personalizado
Datos de aplicación
Propiedades de material
No se pueden editar los materiales en la biblioteca predeterminada. Para editar un material, cópielo primero a una biblioteca personalizada.
Tipo de modelo: Isotrópico elástico lineal
Guardar tipo de modelo en la biblioteca
Unidades: SI - N/mm^2 (MPa)
Categoría: Plástico
Nombre: PETG CREALITY 40% INFILL
Criterio de fallos predeterminado: Tensión de von Mises máx.
Descripción: -
Origen:
Sostenibilidad: No definido
Seleccionar...
Propiedad | Valor | Unidades
Módulo elástico | 2000 | N/mm^2
Coeficiente de Poisson | 0.38 | N/D
Módulo cortante | 800 | N/mm^2
Densidad de masa | 660.65 | kg/m^3
Límite de tracción | 75 | N/mm^2
Límite de compresión | | N/mm^2
Límite elástico | 55 | N/mm^2
Coeficiente de expansión térmica | 60 | /K
Conductividad térmica | 0.2256 | W/(m·K)
Calor específico | 1386 | J/(kg·K)
Acceder a más materiales desde el portal web de materiales de SOLIDWORKS
Agregar...
Guardar
Config...
Aplicar
Cerrar
Ayuda

Figura 41. Propiedades físicas del modelo según el material de PETG.

Una vez configurado el nuevo material con las propiedades del PETG, se hizo el análisis en cada pieza configurando su cara fija (serían las articulaciones dado que la unión es entre los eslabones) y el torque correspondiente al motor que aplica justo en ese eslabón en particular, debido a que tanto el giro en los dos puntos de un solo eslabón bien pueden ser en ambas direcciones en cada uno, se consideró el peor de los casos para reducir a un solo análisis por pieza (de lo contrario serian 4 por parte, pero analizando el caso más crítico entonces se da por hecho que los otros casos son mejores), que se puede ver en el recuadro azul: el torque en su giro, así

como los datos registrados de torque en el recuadro rojo.

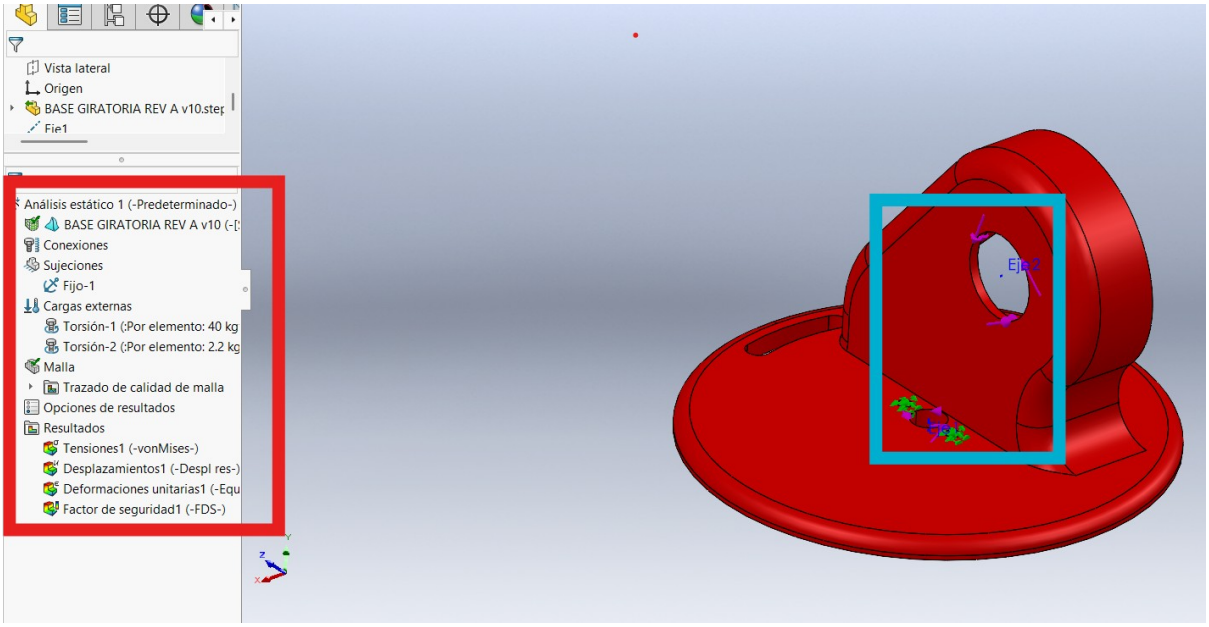


Figura 42. Configuración del esfuerzo aplicado del motor para el análisis.

Ya con esto se realizó el análisis después de crear un mallado, el programa utiliza el método de elementos finito para eso, pero el programa lo realiza en el entorno de análisis estático, y entrega el resultado, solo que la escala en la visualización está aumentada para ver el resultado, un ejemplo de esto es al visualizar el supuesto desplazamiento de una pieza, debido a la escala se ve una versión exagerada del resultado, pero al observar la escala podemos verificar que no hay un problema considerable.

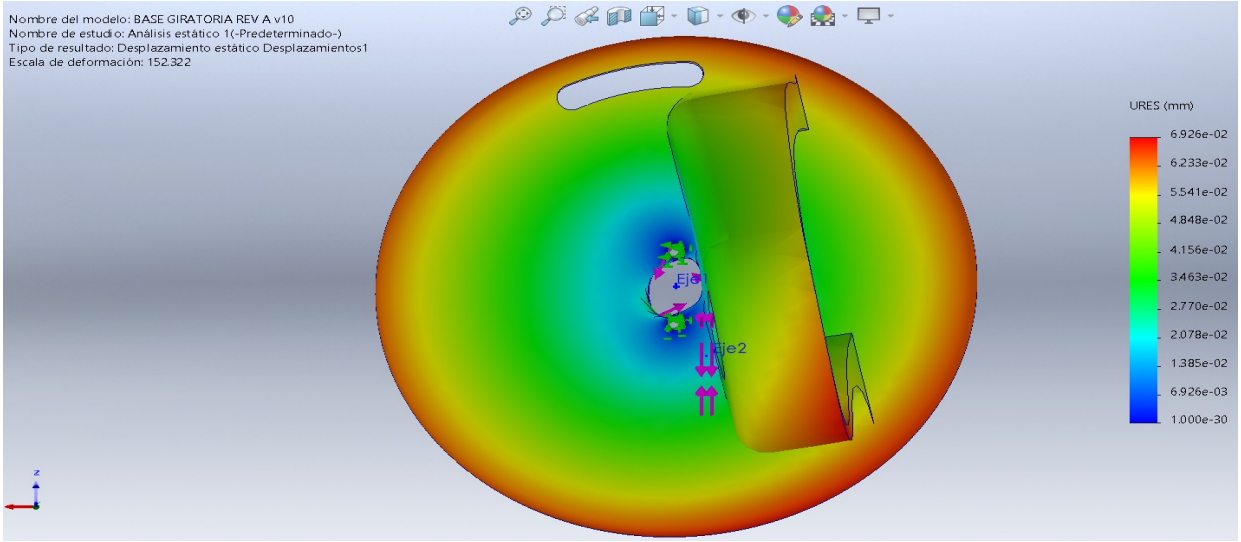


Figura 43. Resultado análisis de esfuerzos.

Una vez ajustada la escala de visualización podemos ver el resultado del factor de seguridad, este valor compara el esfuerzo máximo en la pieza y la resistencia máxima del material, este aspecto lo hizo el factor a evaluar las piezas para su uso. Generalmente un factor de seguridad mayor a 1 nos dice en teoría que una pieza no se romperá, y uno menor a 1 es un riesgo de falla o ruptura, sin embargo, por cuestiones de confiabilidad se recomienda tener un valor dentro de cierto rango según el material o incluso aplicación.

.

Para materiales dúctiles, el factor de seguridad recomendado es de 1.5 a 2 mínimo, mientras que para materiales frágiles de 2 a 3, como estamos tratando con materiales con alta incertidumbre (impresiones 3D) se recomienda un rango entre 3 a 4 por variaciones en propiedades y posibles defectos de fabricación. Para la base giratoria del brazo que se aprecia en la siguiente imagen se observa que toda la pieza en general tiene un factor mayor a 2, si bien no está dentro del margen de materiales de alta incertidumbre, aún cumple para materiales dúctiles y es mayor al mínimo, siendo aceptable para la aplicación didáctica del brazo (el análisis estático de todas las piezas se encuentra en el anexo 2).

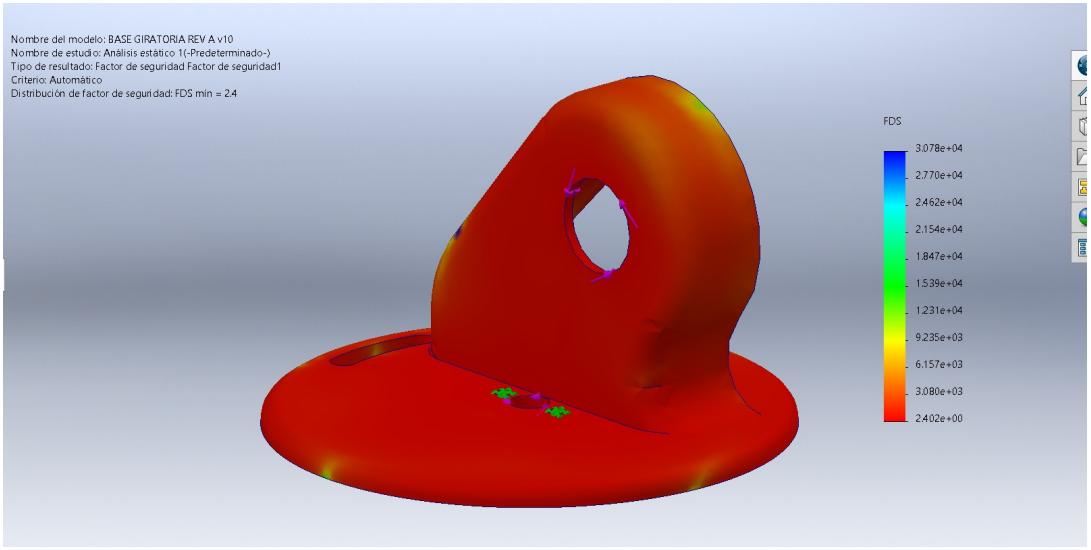


Figura 44. Resultado análisis de desplazamientos.

3.3 Arquitectura del sistema de control electrónico.

En esta parte del proyecto se describe la arquitectura electrónica conceptual que define la forma en que los componentes principales del brazo robótico —servomotores, microcontrolador y elementos de interconexión— se relacionan para lograr un control preciso del movimiento.

Esta fase del proyecto no contempla aún el cableado físico definitivo; en cambio, se establece un análisis lógico y estructural de cómo deben realizarse las conexiones para asegurar la compatibilidad, funcionalidad y eficiencia del sistema. Esta planificación resulta crítica para evitar errores en etapas posteriores del ensamble, reducir riesgos de daño a los componentes y facilitar el desarrollo del software de control.

Selección de actuadores:

El modelo preliminar del brazo robótico condiciona la elección de los servomotores utilizados, ya que dicho diseño ya contemplaba los espacios físicos para el montaje de actuadores estándar. Con base en las necesidades

de torque en cada articulación, se seleccionaron dos tipos distintos de servomotores:

MG995 (torque alto): Este modelo fue asignado a las dos articulaciones que requieren mayor esfuerzo mecánico: la cintura (rotación de la base) y el hombro. Se trata de un servomotor con engranajes metálicos, capaz de suministrar hasta 40 kg·cm a 6V, lo que permite manejar el peso del brazo extendido y los efectos de palanca de las secciones superiores. Su carcasa es robusta y su compatibilidad con soportes estándar lo hace ideal para este tipo de proyectos.

MOT-110 (micro servo de Steren): Se eligió para las articulaciones del codo, la muñeca y el gripper, donde el esfuerzo requerido es considerablemente menor. Este micro servo es compacto y liviano, con un torque nominal de aproximadamente 2.2 kg·cm, lo suficiente para garantizar movimientos precisos sin afectar la estructura general del brazo. Además, su tamaño es el adecuado para integrarse en los alojamientos diseñados en el modelo CAD preliminar.

La correcta asignación de estos actuadores no solo considera el torque, sino también la compatibilidad física con los eslabones del brazo, evitando tener que rediseñar todo el sistema mecánico y solo adaptar las juntas del robot. Esta decisión estratégica optimiza tiempo de desarrollo y recursos materiales.

Microcontrolador: Arduino Nano

El controlador seleccionado para este proyecto es un Arduino Nano, por su tamaño compacto, facilidad de programación y cantidad adecuada de pines para una primera implementación funcional. Dispone de hasta 6 pines PWM (Pulse Width Modulation), lo cual permite controlar directamente hasta seis servos mediante señales digitales de modulación por ancho de pulso.

La elección del Nano también se justifica por su bajo consumo energético, compatibilidad con múltiples entornos de desarrollo (como Arduino IDE y Lab View) y su facilidad para ser integrado en prototipos mediante placas de prueba (protoboard) o shields (tablillas) diseñados a la medida del proyecto.

Consideraciones de alimentación eléctrica

Uno de los aspectos más importantes en la planeación de la arquitectura es el manejo adecuado de la energía. Si bien el Arduino Nano puede alimentarse por USB o una entrada externa de 5V, no es capaz de suministrar la corriente necesaria para alimentar múltiples servomotores simultáneamente. Cada MG995 puede consumir picos de más de 2 A al moverse con carga, mientras que los MOT-110 también tienen requerimientos importantes, especialmente si todos se activan al mismo tiempo.

En este proyecto se emplean cinco servomotores, de los cuales dos son de alto torque modelo MG995, y los tres restantes (incluyendo el gripper) son microservos modelo MOT-110 de la marca Steren.

Todos los servomotores están conectados en paralelo, es decir, cada uno cuenta con su propio cable de alimentación positiva (V+) y negativa (GND) conectados directamente a las líneas comunes de 5V y GND, respectivamente. Esta configuración asegura que cada servo reciba el mismo voltaje de 5V, condición indispensable para su correcto funcionamiento.

Una conexión en serie no sería adecuada, ya que implicaría dividir el voltaje entre los dispositivos, reduciendo drásticamente el voltaje disponible para cada uno e impidiendo su operación. Además, los servos requieren corrientes distintas, por lo que deben estar alimentados en paralelo para funcionar de forma independiente.

Cálculo de Corriente Total y Caída de Voltaje

Se realizó un análisis teórico considerando el consumo máximo estimado de cada tipo de servo:

Modelo de Servo	Cantidad	Corriente Máxima Aproximada	Corriente Total
MG995	2	2.0 A	4.0 A
MOT-110	3	0.8 A	2.4 A
Total	5	—	**6.4–7.2 A** (pico)

Tabla 3. Cálculos de consumo de corriente eléctrica de los motores del robot.

La corriente fluye por un cable común de alimentación (5V y GND), cuya resistencia eléctrica depende del calibre y longitud del conductor. Con un cable AWG 20 (resistencia aproximada de 0.066 Ω), la caída de voltaje en condiciones de corriente máxima se estima como:

$$V_{caída} = I \cdot R = 7.2\,A \cdot 0.066\,\Omega \approx 0.475$$

Por tanto, el voltaje real que reciben los servos sería aproximadamente:

$$V_{servos} = 5\,V - 0.475\,V = 4.52\,V$$

Este valor está dentro del rango de funcionamiento permitido por los servos (típicamente 4.5–6 V). Además, en condiciones normales de operación (no en carga pico), la corriente total es menor, y por ende la caída de tensión se reduce aún más (aproximadamente 0.3–0.4 V), garantizando una operación estable.

Por esta razón, la alimentación de los servos debe realizarse mediante una fuente externa dedicada de 5 V DC, capaz de entregar al menos 7A para mantener la estabilidad en la operación.

Esta fuente debe tener su tierra (GND) conectada en común con la del Arduino, para que las señales PWM sean correctamente reconocidas por los

servos.

El esquemático o diagrama electrónico del proyecto se puede ver a mayor profundidad en el anexo 3 del presente documento.

Distribución lógica de conexiones

En este diseño preliminar, se utilizarán cables tipo jumper para protoboard, ideales para el desarrollo y pruebas iniciales. Esto permite una conexión rápida, flexible y modificable, esencial durante la etapa de ajustes. Los cables se organizan para evitar interferencias y permitir una fácil trazabilidad del sistema.

Las señales de control de cada servo se conectarán a pines digitales del Arduino con capacidad PWM (para el proyecto, D3, D5, D6, D9, D10 y D11), garantizando que el microcontrolador pueda generar las señales necesarias para mover con precisión cada actuador. Cada señal se enviará individualmente desde el Arduino a la línea de control del servo correspondiente.

Reflexiones preliminares sobre la arquitectura

Diseñar correctamente esta arquitectura electrónica desde una perspectiva lógica es indispensable para el éxito del proyecto. Errores comunes como alimentar servos desde el Arduino, no compartir la tierra, o no usar pines PWM para el control pueden ocasionar fallas intermitentes o daño directo a los componentes.

Además, esta planeación ayuda a prever el tipo de estructura física del cableado, si será ordenado, si permitirá mantenimiento fácil, y si será compatible con una futura placa PCB o shield personalizada para consolidar el prototipo final.

Este enfoque modular, con distinción entre etapa de señal (Arduino) y etapa de potencia (alimentación de servos), sienta las bases para un sistema estable, seguro y escalable.

En esta sección se describe la arquitectura electrónica implementada para controlar el brazo robótico. El sistema está compuesto principalmente por un microcontrolador Arduino Nano, cinco servomotores (tres estándares y dos de mayor torque), una fuente de alimentación externa, y una distribución ordenada de conexiones para minimizar interferencias y facilitar el mantenimiento.

Dado que los servomotores requieren una corriente mayor a la que puede entregar directamente el microcontrolador, se optó por alimentar los motores desde una fuente externa de 5V–6V regulada, común a todos los servos, mientras que el pin de control de cada uno se conecta directamente a un pin digital del Arduino Nano. Se tuvo especial cuidado en compartir la masa común (GND) entre la fuente y el microcontrolador para evitar diferencias de potencial que podrían dañar los componentes.

Los servomotores fueron organizados en función de su carga estimada: los tres de 2.2 kg·cm (Steren) fueron asignados al gripper, muñeca y codo, mientras que los servomotores de mayor torque (tipo MG995 o similar) se destinaron a la base y al hombro. El orden de conexión de los pines fue cuidadosamente planificado para facilitar la programación y depuración del sistema.

Asimismo, se verifica la funcionalidad del sistema a nivel de hardware, comprobando que todos los componentes electrónicos respondan a las señales generadas desde el microcontrolador. Este apartado también puede incluir un pequeño análisis del esquema eléctrico general y del uso de placas de expansión, fuentes de alimentación externas o módulos de comunicación si se utilizaron.

3.4 Firmware inicial del control de movimiento.

Programación del arduino.

En esta etapa se desarrolló e implementó el firmware que permite la recepción e interpretación de comandos desde una interfaz externa hacia el microcontrolador (Arduino Nano), con el fin de controlar el movimiento del brazo robótico.

El programa cargado al microcontrolador fue diseñado para actuar como intermediario entre la interfaz gráfica del usuario y los servomotores que accionan las articulaciones del brazo. Para esto, se emplea comunicación serial, a través de la cual se envían caracteres simples —como K para abrir el gripper y L para cerrarlo— así como secuencias de ángulos correspondientes a cada grado de libertad.

El código interpreta estos comandos y ajusta la posición de los motores en consecuencia, realizando los movimientos deseados. Además, esta implementación permite verificar la correcta integración de los componentes electrónicos y mecánicos, validando la funcionalidad del sistema antes de integrar rutinas más complejas o control visual.

En esta etapa se lleva a cabo la instalación del microcontrolador (Arduino Nano), que constituye el núcleo del sistema de control del brazo robótico. Se conectan los servomotores a los pines de salida PWM correspondientes, se suministra alimentación regulada y se conecta la interfaz de comunicación serial para futura programación.

Puente de comunicación entre LabView y Arduino: Lifa Base

Para lograr la comunicación entre el entorno de desarrollo gráfico

LabVIEW y el microcontrolador Arduino Nano, se utilizó LIFA Base (LabVIEW Interface for Arduino). Esta herramienta actúa como un puente de comunicación, permitiendo que LabVIEW envíe y reciba datos hacia y desde el Arduino a través de comandos específicos. Al funcionar como un enlace intermedio, Lifa Base traduce las instrucciones generadas en LabVIEW a un formato comprensible para el microcontrolador, facilitando así el control de entradas y salidas digitales, señales PWM y lectura de sensores. Gracias a esta integración, es posible manejar el prototipo del brazo robótico desde una interfaz gráfica, sin necesidad de reprogramar el Arduino constantemente.

Primero que nada, debemos cerciorarnos de contar con la extensión de Linx, para detectar el dispositivo de arduino en el software de Lab View. Esto se puede verificar desde la aplicación de VI Package Manager.

Figuras 45 y 46.Icono del VI Package Managerx.

Dentro del programa, podemos buscar la palabra “Linx”, debe aparecer la extensión Digilent Linx (Control Arduino).

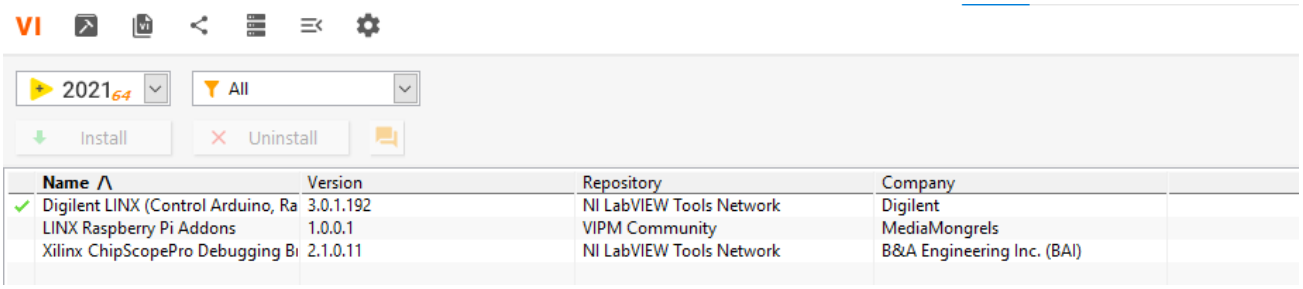


Figura 47. Digilent LINX extension.

Ahora, para establecer la comunicación, debemos tener un dispositivo arduino, conectado a la computadora. Una vez conectado y verificado que el puerto está detectado por la computadora, navegaremos a la sección en Lab View de Tools >> MakerHub >> LINX >> Linx Firmware Wizard.

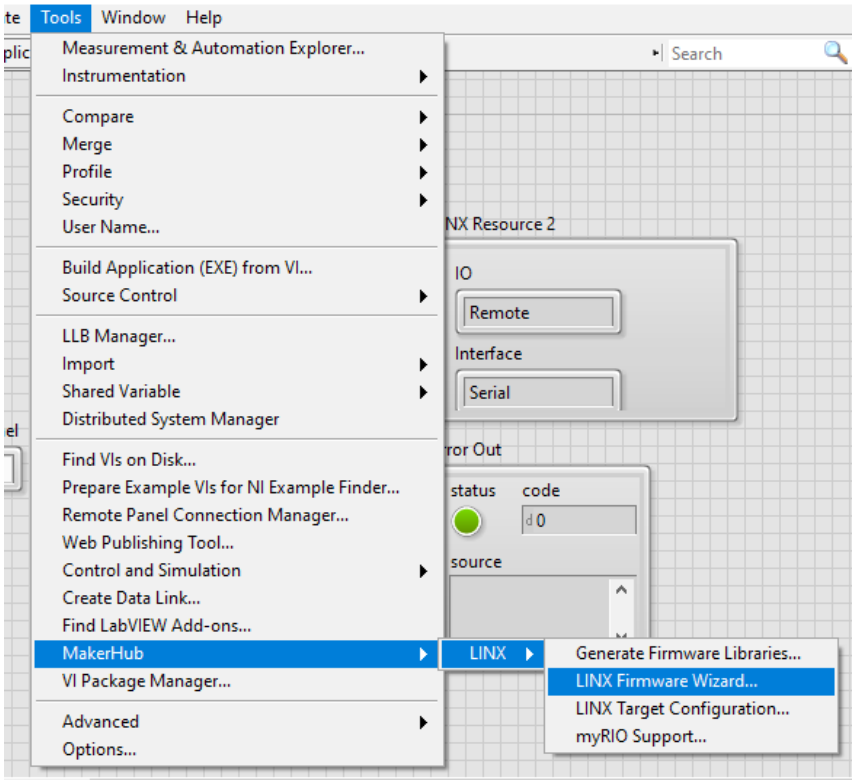


Figura 48. Ubicación de la extensión dentro de Lab View.

Esto desplegará la ventana emergente debajo, a la cual le adecuamos el Device Type, al que se utilice, para este caso: arduino nano. Dar en continuar

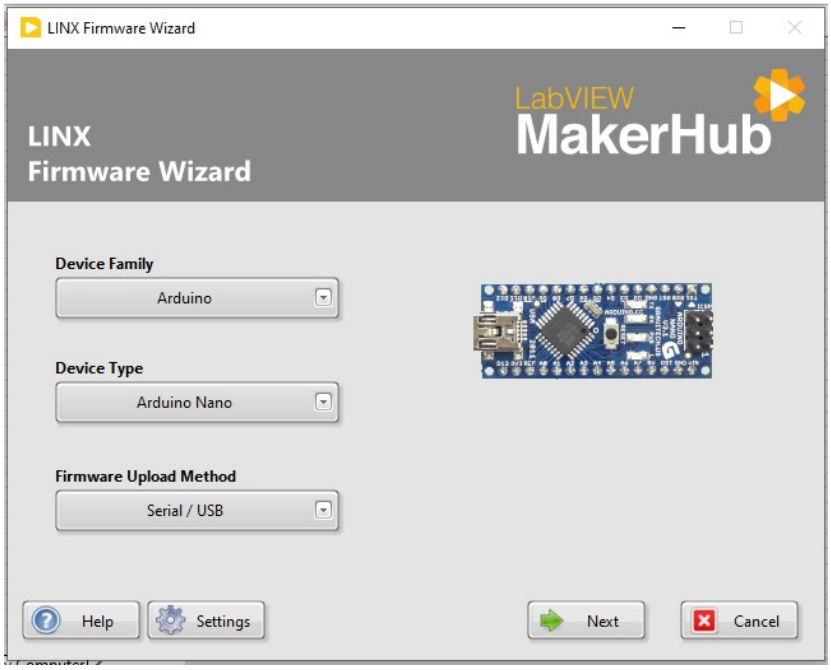


Figura 49. Menú Principal de LINX Firmware Wizard.

Seleccionar el puerto COM al que tenemos conectado el dispositivo, y dar en continuar.

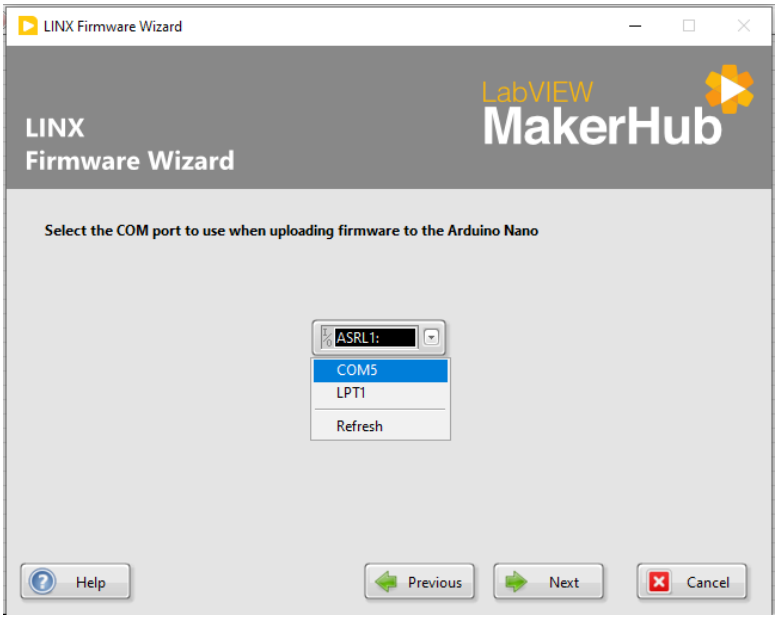
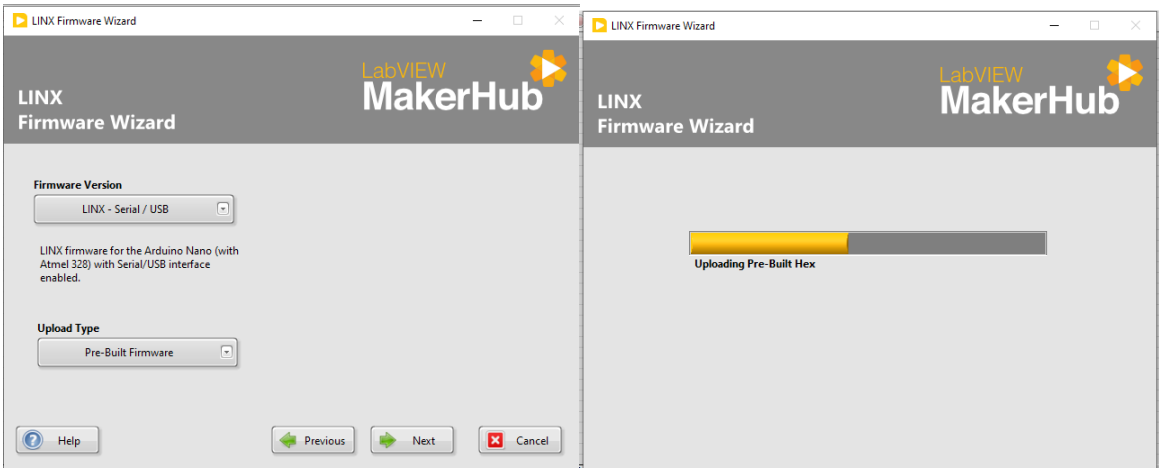


Figura 50. Selección de puerto disponible.

Lo siguiente será esperar a que el firmware wizard permita la comunicación con la placa de arduino. Deberá aparecer una imagen donde el

proceso se haya completado.



Figuras 51 y 52. Seguimiento de comunicación entre Lab View y el microcontrolador.

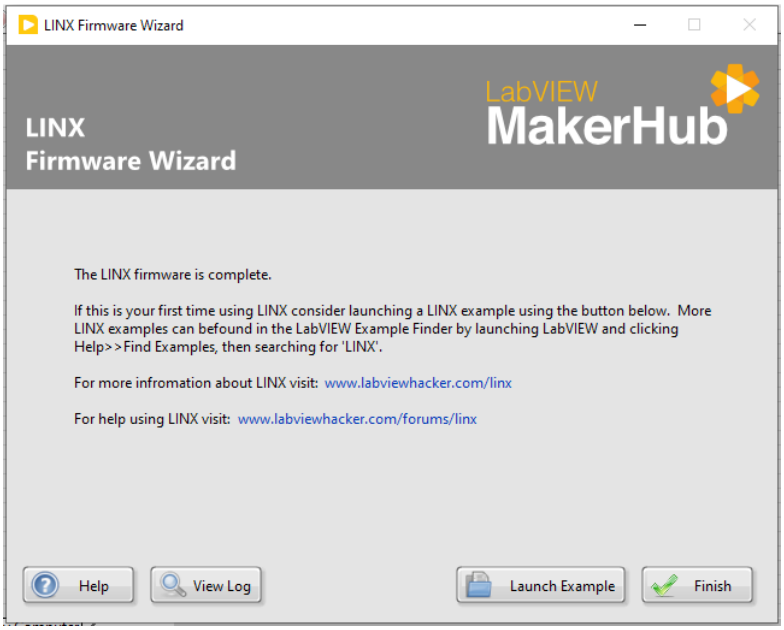

LINX Firmware Wizard
LINX
Firmware Wizard
LabVIEW
MakerHub
The LINX firmware is complete.
If this is your first time using LINX consider launching a LINX example using the button below. More LINX examples can befound in the LabVIEW Example Finder by launching LabVIEW and clicking Help>>Find Examples, then searching for 'LINX'.
For more infromation about LINX visit: www.labviewhacker.com/linx
For help using LINX visit: www.labviewhacker.com/forums/linx
Help
View Log
Launch Example
Finish

Figura 53. Respuesta satisfactoria del microcontrolador.

El VI Init (Initialize Arduino.vi) es la puerta de entrada que configura y establece la comunicación entre LabVIEW y Arduino a través del puerto serial. Este VI se encarga de preparar el entorno para que los comandos posteriores (como Servo Write, Digital Write, Analog Read, etc.) puedan ser enviados correctamente al microcontrolador.

Al ejecutar el VI Init, se realiza lo siguiente:

1. Se abre el puerto serial especificado por el usuario (por ejemplo, "COM4").
2. Se configura la velocidad de transmisión (baud rate), típicamente a 115200 baudios, que debe coincidir con la velocidad del firmware cargado en el Arduino.
3. Se prepara el canal para enviar y recibir datos con buffers adecuados.

Una vez abierto el puerto, Init envía un paquete especial de inicialización, que generalmente contiene una instrucción como "ping" o "identifícate".

El Arduino con el firmware de LIFA Base responde con un paquete de confirmación, que incluye:

- Versión del firmware LIFA.
- Tipo de placa conectada (por ejemplo, "Nano", "Uno", etc.).
- Información del estado inicial de pines.

Esto asegura que el Arduino esté listo y "escuchando" correctamente. En términos de bloques de LabVIEW, el VI Init contiene:

- Un bloque de configuración de VISA Serial Port.
- Un envío de un paquete de handshake a través del puerto.
- Un Read VISA que espera respuesta del Arduino para verificar compatibilidad.
- Un nodo de salida llamado Device Reference: este "handle" o referencia se usa en los siguientes VIs para que todos sepan a qué microcontrolador están hablando.

El VI Init debe ejecutarse una sola vez al inicio del programa, antes de cualquier otro comando que involucré al Arduino. Si no se realiza esta inicialización, los demás VIs no podrán comunicarse ni enviar instrucciones válidas.

Por lo general, el flujo de trabajo es así:

Paquete que envía Init a Arduino:

[0xFF] [CMD: GET_FIRMWARE_VERSION] [0x00] [0xF0]

Arduino responde:

[0xFF] [RESPONSE: VERSION] [Major: 1] [Minor: 6] [0xF0]

Este tipo de confirmación asegura que ambas partes "hablen el mismo idioma".

Aunado a esto, también debemos de configurar la comunicación entre LabView y Arduino dentro de LabView, esto para garantizar el correcto envío de señales, para esto se realizan los siguientes pasos dentro de Lab View, con la extensión de Linx.

El VI Init no es un simple bloque de inicio: es un componente crítico de configuración que asegura que LabVIEW y Arduino establezcan una comunicación robusta, sincronizada y validada antes de ejecutar cualquier operación de control o adquisición de datos.

Una vez inicializado correctamente, puedes controlar pines, leer sensores y mover actuadores de forma confiable desde tu interfaz gráfica, con una arquitectura en tiempo real basada en estructuras de control como While Loops y Case Structures.

3.5 Armado y cableado del prototipo.

Una vez concluido el proceso de impresión 3D de las piezas del modelo preliminar, se procedió con el armado físico del prototipo del brazo robótico. Esta etapa consistió en el ensamblaje cuidadoso de cada uno de los eslabones y componentes estructurales, respetando las orientaciones y tolerancias definidas en el diseño CAD. Fue fundamental verificar que las piezas encajaran adecuadamente entre sí, especialmente en las zonas de unión de articulaciones, donde se requería un alineado preciso para evitar limitaciones en el rango de movimiento o generación de esfuerzos no deseados.

Durante el proceso de armado, se instalaron también los servomotores seleccionados para las distintas articulaciones. Los modelos MG995 fueron integrados en las articulaciones de mayor exigencia mecánica (cintura y hombro), debido a su mayor torque y robustez, mientras que los servomotores MOT-110 se asignaron a las articulaciones de menor carga (codo, muñeca y gripper), por su tamaño reducido y compatibilidad dimensional con el diseño. Esta distribución permitió equilibrar el peso y facilitar la integración mecánica sin comprometer la funcionalidad.

Posteriormente, se llevó a cabo el cableado del sistema eléctrico, conectando los servomotores a los pines PWM del microcontrolador (Arduino Nano) a través de una protoboard intermedia. En esta fase, se emplearon

cables tipo jumper para facilitar las conexiones y permitir una organización limpia del cableado. Se prestó especial atención a la alimentación eléctrica, optando por una fuente de 5V externa en lugar de depender del regulador del Arduino, ya que este último no es capaz de suministrar la corriente necesaria para el funcionamiento simultáneo de múltiples servos. Se estableció una conexión común a tierra entre la fuente y el microcontrolador, con el fin de asegurar una referencia eléctrica compartida que evitará interferencias o fallos de comunicación.

Esta actividad fue clave no solo para integrar físicamente los subsistemas mecánico y electrónico, sino también para identificar ajustes necesarios en el diseño estructural o distribución interna del prototipo. A partir del armado y cableado inicial, se detectaron algunas dificultades relacionadas con la posición de ciertos motores, la longitud de los cables, y el acceso a los pines del microcontrolador, lo cual permitió una mejora posterior en la disposición interna y la funcionalidad general del brazo.

Posteriormente, realizamos el cableado de los actuadores y sensores, asegurando una organización ordenada del sistema eléctrico que facilite el mantenimiento y reduzca interferencias. Se emplean conectores, terminales y, en su caso, técnicas de soldadura para garantizar conexiones seguras. También se cuida la correcta polaridad y distribución de energía en todo el prototipo.

Se identificaron diversas áreas de mejora relacionadas con la resistencia mecánica, estabilidad, integración de componentes y capacidad de movimiento. A partir de esta experiencia, se decidió rediseñar el modelo en el software Fusion 360, tomando en cuenta los siguientes factores:

- Incorporación de rieles para mejorar la estabilidad del brazo y facilitar el recorrido de los cables.
- Refuerzo de zonas de unión que presentaban fragilidad con el PLA.

- Redimensionamiento de piezas críticas, optimizando geometrías para reducir fricción y mejorar el alcance de movimiento.
- Adaptación de espacios internos para alojar correctamente los servomotores y el cableado.

3.6 Rediseño del modelo

Al tener en cuenta el modelo físico, se hicieron varias modificaciones para optimizar el modelo inicial, para esto primero se realizó una versión propia de la pieza conservando sus dimensiones relevantes, con tal de poder trabajarlo con menos dificultad, las siguientes imágenes son una comparación del proceso que se desarrolló en todas las piezas (el rediseño de todas las piezas se encuentra en el anexo 5).

Figura 54. Diseño inicial del brazo.

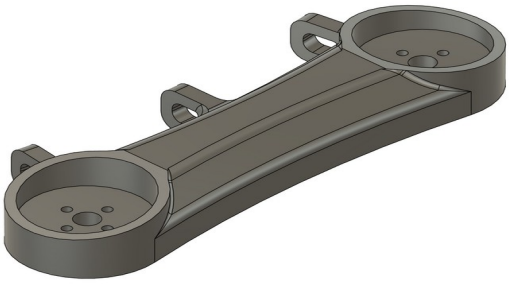


Figura 55. Diseño final del brazo.

En este eslabón en particular, así como en los demás eslabones del brazo se tiene una mejora en la adición de guías exteriores para el cableado de los motores, también para todo el brazo se mejoró la distancia entre eslabones para mantener una distancia aceptable de tolerancia para contacto, dado que es preferible evitar la constante fricción entre piezas se dejó un pequeño margen de separación, pero mucho mejor al que presentaba el modelo inicial. También se diseñó una carcasa para proteger el microcontrolador para evitar un impacto accidental tanto del brazo como de algún factor externo.

Dado que se tiene contemplado el uso de motores con una capacidad considerable para la resistencia del plástico, se optó por cambiar el material usado en el prototipo, para mantener una relación calidad-precio que no desbalance la integridad óptima del brazo a su desempeño se optó por elegir PETG, ya que presenta muchas mejoras mecánicas al PLA como mayor

resistencia a la tensión, no presenta solubilidad ante los químicos convencionales (el PLA se disuelve en acetona), ya pesar de que su uso será en laboratorio, no presenta una degradación tan rápida al sol como la que presenta el PLA (dado que este es derivado de materias primas naturales). Ya con las piezas en su versión final se hizo nuevamente el ensamble en su versión final, incluyendo una base para el proyecto.

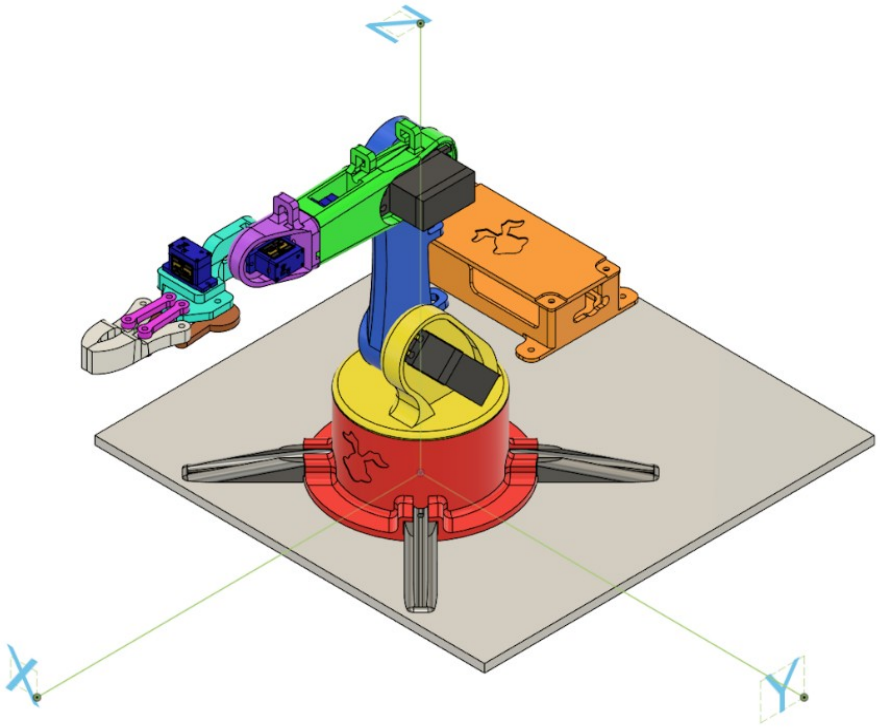


Figura 56. Ensamble con los modelos finales del robot.

La selección de los componentes electrónicos fue realizada considerando la compatibilidad con el diseño mecánico, la capacidad de carga requerida por cada articulación, la facilidad de control y la disponibilidad de materiales. Si bien el modelo preliminar ya contemplaba el uso de servomotores, se llevó a cabo una evaluación de sus características para validar su idoneidad.

Para las tres articulaciones que requieren menor torque (muñeca, codo y gripper), se utilizaron servomotores de 40 kgf·cm, específicamente del tipo

comúnmente comercializado por Steren, adecuados para tareas de precisión media y bajo requerimiento de fuerza. Estos servomotores ofrecen una relación adecuada entre costo, tamaño y capacidad de carga, siendo ideales para movimientos controlados en aplicaciones educativas.

En el caso de las articulaciones que soportan mayor carga, como la base (cintura) y el hombro, se optó por servomotores más robustos, capaces de soportar mayores exigencias de torque y garantizar estabilidad durante el funcionamiento.

3.7 Recuento de materiales

Después de haber hecho el rediseño del brazo para su optimización, se hizo un recuento de todos los materiales requeridos para construir un brazo funcional con todas las mejoras hechas en el modelo y de esa forma poder recabar todos los materiales necesarios con el fin de construir el modelo desde cero, incluso para poder replicarlo.

Para poder hacer una suma del plástico utilizado en las impresiones 3D, si bien se pueden pesar las piezas, no sería muy exacto dado que haremos el cambio de PLA a PETG (Tienen densidades diferentes, mientras el PLA genérica se tiene en 1.25 gramos por centímetro cúbico, mientras que el PETG puede variar de 1.27 a 1.30 gramos por centímetro cúbico, el programa slicer maneja el valor más bajo de forma predeterminada), por lo que haremos uso del software slicer de Creality Print nuevamente para considerar el nuevo material a utilizar. Se procedió a crear los códigos G para las piezas rediseñadas mientras el programa nos da un previo del tiempo y material usado.

La impresión se dividió en seis bandejas diferentes de impresión, la idea fue equilibrar el tiempo total estimado entre las bandejas lo mejor posible, dándonos un total estimado de 400 gramos aproximadamente, en un lapso de tiempo continuo de 26 horas (Sumando todo el tiempo en que la impresora estará fabricando las piezas).

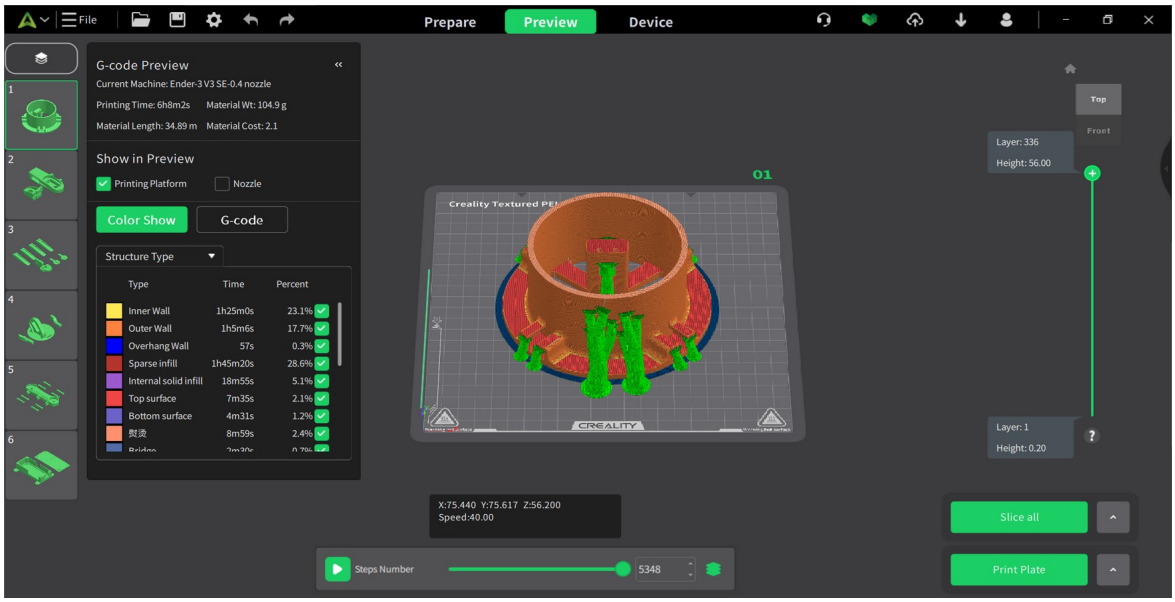


Figura 57. Bandeja de impresión 1.

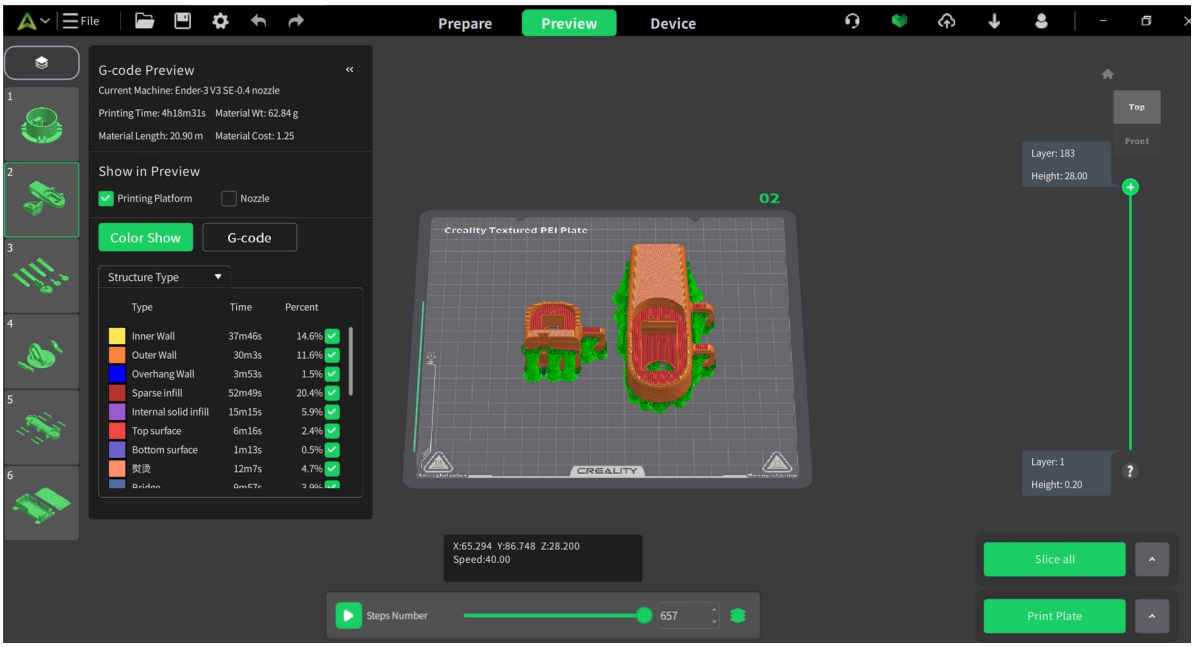


Figura 58. Bandeja de impresión 2.

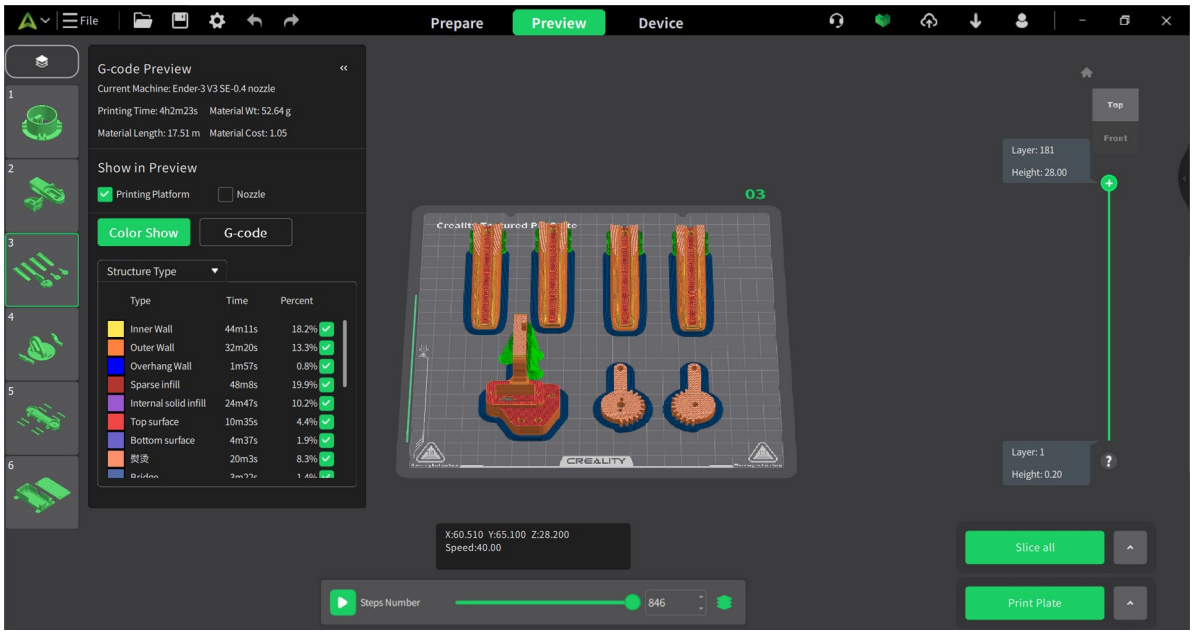


Figura 59. Bandeja de impresión 3.

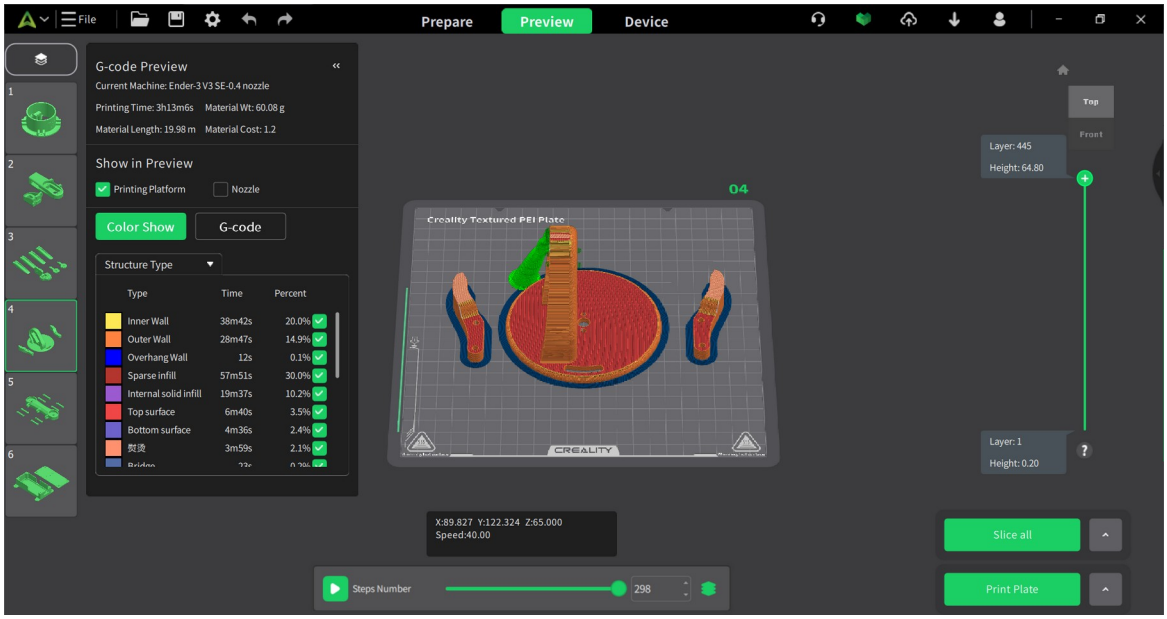


Figura 60. Bandeja de impresión 4.

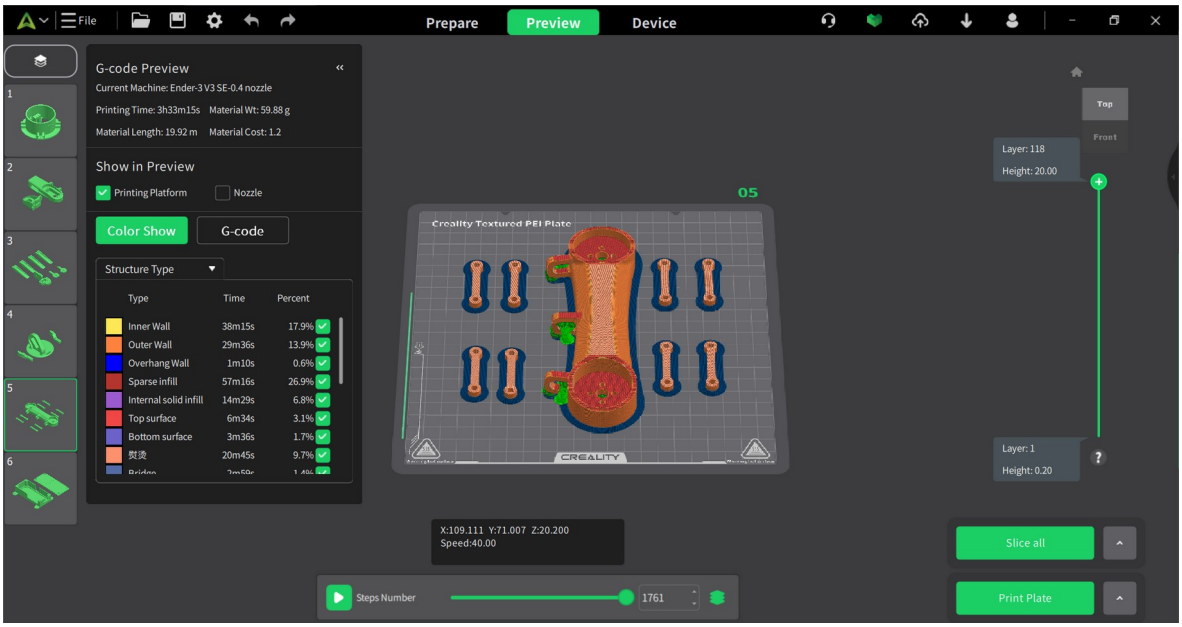


Figura 61. Bandeja de impresión 5.

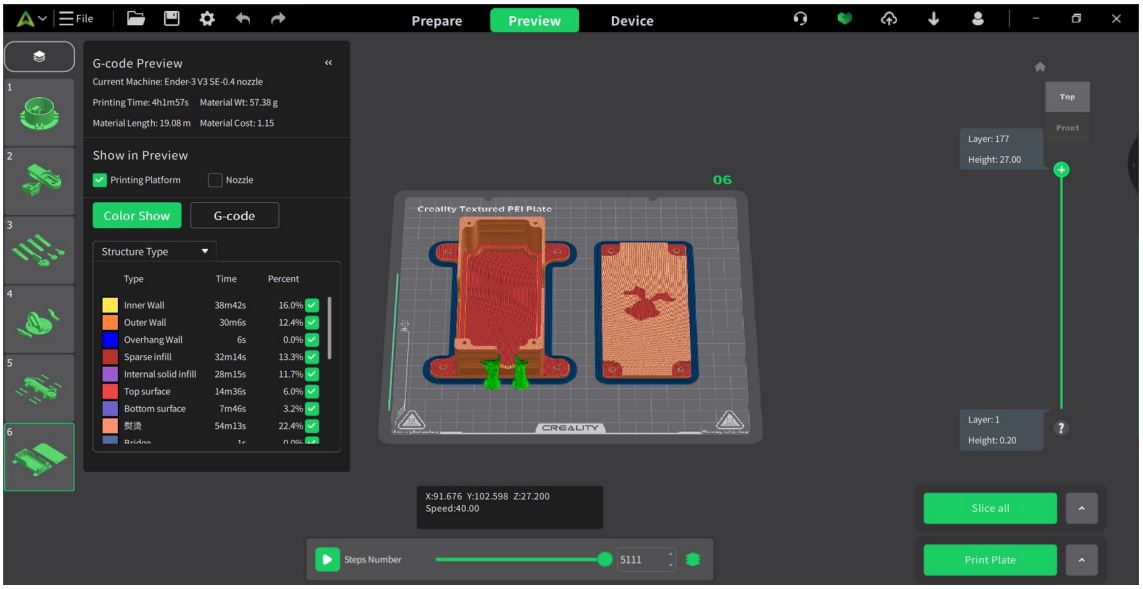
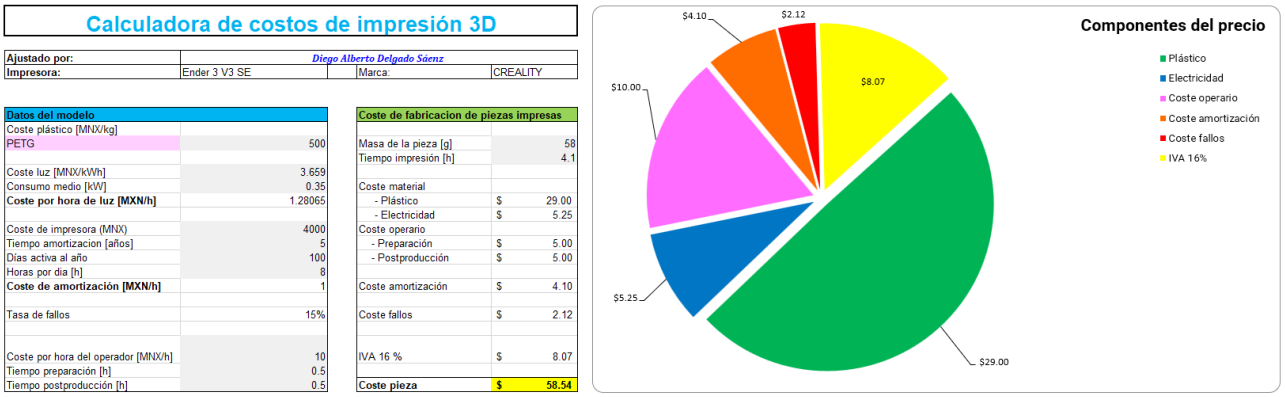


Figura 62. Bandeja de impresión 6.

El material para impresión 3D se distribuye normalmente en rollos de 1 kilogramo, por lo que solo se necesitará una unidad para construir las piezas e incluso dejar material para reemplazar por lo menos 1 vez todas las piezas en caso de necesitar su reemplazo.

Cabe recalcar que es posible realizar una cotización previa al costo de

la impresión utilizando un cotizador diseñado para considerar el uso de material, energía, tiempo, entre otros aspectos, pero el precio de una pieza puede variar mucho si se imprime únicamente esa pieza a si se imprime en conjunto con otras en la misma bandeja de impresión como en este caso, lo cual fue de utilidad dividir todas las piezas entre varias bandejas.



Calculadora de costos de impresión 3D

Ajustado por:	*Diego Alberto Delgado Sáenz*		
Impresora:	Ender 3 V3 SE	Marca:	CREALITY

Datos del modelo	
Coste plástico [MNX/kg]	
PETG	500
Coste luz [MNX/kWh]	3.659
Consumo medio [kW]	0.35
Coste por hora de luz [MXN/h]	1.28065
Coste de impresora (MNX)	4000
Tiempo amortizacion [años]	5
Días activa al año	100
Horas por dia [h]	8
Coste de amortización [MXN/h]	1
Tasa de fallos	15%
Coste por hora del operador [MNX/h]	10
Tiempo preparación [h]	0.5
Tiempo postproducción [h]	0.5

Coste de fabricacion de piezas impresas		
Masa de la pieza [g]		58
Tiempo impresión [h]		4.1
Coste material		
- Plástico	$	29.00
- Electricidad	$	5.25
Coste operario		
- Preparación	$	5.00
- Postproducción	$	5.00
Coste amortización	$	4.10
Coste fallos	$	2.12
IVA 16 %	$	8.07
Coste pieza	**$**	**58.54**

Figura 63. Descripción de costos en materiales para un brazo robótico.

El recuento del material para construir un brazo robótico se encuentra en el anexo 6.

3.8 Desarrollo de interfaz gráfica en Lab View

Con el objetivo de ofrecer una alternativa visual para la supervisión y control del brazo robótico, se diseñó una interfaz gráfica utilizando el entorno de desarrollo LabVIEW. Esta plataforma fue seleccionada por su capacidad para crear interfaces intuitivas y su compatibilidad con la comunicación serial de nuestro microcontrolador, lo cual facilita la interacción con el microcontrolador del sistema.

La interfaz permite al usuario visualizar los ángulos de cada articulación del brazo, enviar comandos básicos (como abrir o cerrar el gripper, dirigir posiciones, movimiento libre del robot, generar secuencias), y monitorear el estado general del sistema. Se incorporaron indicadores, controles deslizantes y botones virtuales para proporcionar una experiencia de uso sencilla y clara, especialmente pensada para fines didácticos para la clase de robótica.

La programación en LabVIEW se estructuró como una máquina de estados, que desplegará un menú al usuario con 3 tipos de apartados, los cuales podríamos asemejar a las áreas industriales que regularmente se gestionan en las maquiladoras, fábricas, talleres, etc. Estos apartados los enumeramos de la siguiente forma:

- Movimiento libre del robot o sección de mantenimiento: esta sección le permitirá al usuario realizar un movimiento libre integral de las articulaciones del robot.
- Cálculos de posición o sección de ingeniería: esta sección realiza los cálculos de la cinemática directa e inversa del robot, esto para conocer la posición de las articulaciones y las coordenadas en el área de trabajo del robot.
- Secuencias o sección de producción: este apartado le permite al usuario guardar posiciones, así como generar secuencias de seguimiento que el robot pueda realizar automáticamente.

Mediante diagramas de bloques, integrando nodos de lectura y escritura serial, así como estructuras condicionales que interpretan los datos enviados por el usuario y los convierten en comandos entendibles por el firmware del microcontrolador.

Esta interfaz complementa la versión realizada en Visual Studio, ampliando las opciones de control y explorando entornos gráficos de ingeniería ampliamente utilizados.

VI de Menú Principal.

El VI de Menú Principal realiza un menú interactivo a partir de la programación de una máquina de estados en Labview, este menú le permitirá navegar en los diferentes apartados de uso del robot (hay un subvi por cada apartado), siendo 3 los desarrollados para el control del robot.

Figura 64. Front panel del Menú Principal.

Hablando a grandes rasgos, el Front Panel contiene 4 botones que le permitirán mandar a llamar los programas o subvi de cada sección. El *Block diagram* contiene la máquina de estados junto con una estructura de *Wait for Event* que permite trabajar con los botones esperando la elección del usuario.

Sub VI de Mantenimiento: Movimiento Libre del robot

Este Sub Vi como su nombre lo dice, le permite al usuario dirigir las articulaciones del robot a partir de un movimiento libre de cualquier articulación. Contiene un Tab Control en el que se puede seleccionar una articulación y un controlador que le permite especificar el grado de posición en el cual quiere que se encuentre el robot. Dicho controlador solo funciona una vez que el botón de encendido se encuentre activado.

Figura 65. Front Panel Movimiento libre.

Además de esto, se permite seleccionar el puerto de comunicación hacia el microcontrolador y su correspondiente botón de home para volver al Menú Principal del proyecto.

Sub VI de Ingeniería: Cálculo de posiciones

Este Sub Vi le permite al usuario revisar la posición del robot, ya sea partir de sus coordenadas en el plano, o bien, la posición de sus articulaciones. El Front Panel con una sección de cinemática inversa y una de cinemática directa. Dependiendo de la información que tengas y a la que quieras llegar, se puede utilizar uno de los dos casos. Ambos casos se complementan con el resultado de los cálculos teóricos para la posición del robot.

Este menú se complementa con la selección del puerto de

comunicación al microcontrolador, un botón para abrir y cerrar el gripper, y el botón de Home para volver al menú principal.

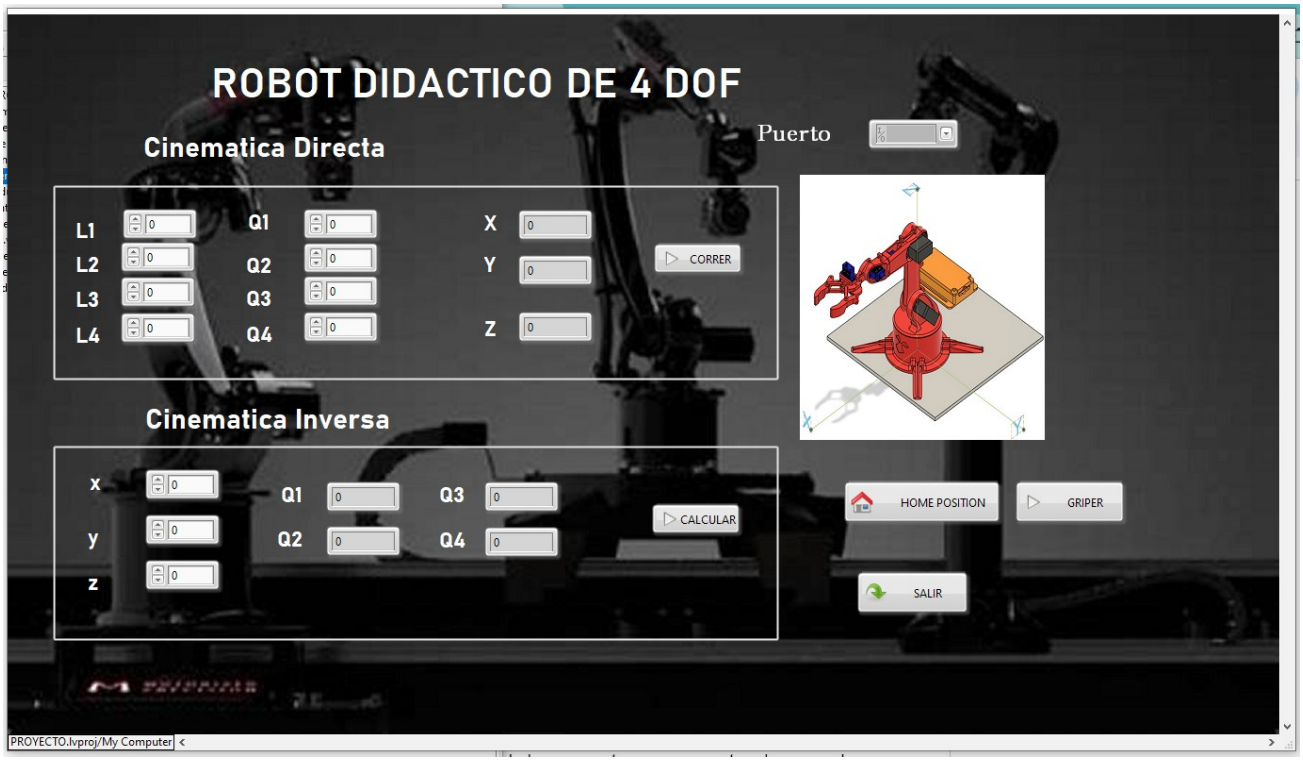


Figura 66. Front Panel de Ecuaciones de posición.

Sub VI de Producción: secuencias.

Este SubVi permite que el robot guarde posiciones y genere secuencias de movimiento, lo que permite definir rutinas de trabajo, testear movimientos de traslado del robot, entre otras muchas aplicaciones de distinto tipo a fin de entender mejor su funcionamiento.

Figura 67. Front panel de secuencias.

Este SubVi contiene un formato de movimiento de posición por articulaciones o coordenadas, lo que nos permite configurar posiciones deseadas, guardarlas y después utilizarlas en una secuencia de movimiento.

3.9 Validación y pruebas funcionales

Una vez preparada la interfaz y rediseñado las diferentes partes del brazo, nuevamente se imprimieron las piezas, recordando que, en esta ocasión al tratarse del modelo final, se optó por utilizar un relleno del 40 por ciento y 3 capas superficiales para darle a los cuerpos una estructura bastante resistente para su uso prolongado. Ya una vez con el material, solo fue cuestión de imprimir las piezas nuevas para realizar el ensamble, el montaje de los motores y el cableado de los mismos.

Una vez ensamblado el brazo robótico y completada la programación del sistema de control, se llevaron a cabo pruebas funcionales para validar el desempeño general del prototipo. Estas pruebas tuvieron como finalidad confirmar la operatividad del sistema mecánico, electrónico y de software bajo condiciones controladas.

Se evaluó el comportamiento de los servomotores en respuesta a los comandos enviados desde la interfaz gráfica, verificando que cada articulación ejecutará los movimientos esperados con precisión y fluidez. Asimismo, se validó la correcta comunicación entre el microcontrolador y la interfaz, asegurando que los datos se transmitieran sin errores.

Durante la validación también se observaron aspectos clave como:

- La alineación y el rango de movimiento de las articulaciones.
- La sincronización entre múltiples servos durante movimientos combinados.
- El tiempo de respuesta del sistema ante los comandos del usuario.
- La funcionalidad del gripper en apertura y cierre.

Las pruebas permitieron detectar y corregir pequeñas fallas de software y de ensamblaje, así como afinar parámetros como los retardos de

movimiento, ángulos límite de seguridad, y rutinas de calibración. Esta fase fue esencial para garantizar que el prototipo cumpliera con su propósito didáctico, permitiendo una manipulación segura, repetible y representativa de un sistema robótico real.

3.10 Evaluación del desempeño del brazo robótico

Con el prototipo completamente ensamblado y funcional, se realizó una evaluación sistemática de su desempeño para medir el cumplimiento de los objetivos técnicos y didácticos establecidos al inicio del proyecto.

Esta evaluación consideró tanto aspectos mecánicos como funcionales y de usabilidad, entre los que se incluyeron:

- Precisión de movimiento: Se verificó si los ángulos ejecutados por cada articulación coincidían con los valores programados.
- Repetitividad: Se realizaron movimientos múltiples con los mismos parámetros para comprobar que los resultados fueran consistentes.
- Velocidad de respuesta: Se midió el tiempo de reacción desde el envío del comando hasta la ejecución completa del movimiento.
- Capacidad de carga: Se probó la resistencia del brazo al levantar objetos de distintos pesos, dentro de los límites esperados.
- Estabilidad estructural: Se observó el comportamiento del brazo ante movimientos prolongados o combinados, verificando que no hubiera deformaciones o fallas mecánicas.
- Interacción con la interfaz gráfica: Se evaluó la facilidad de uso de las interfaces desarrolladas (Visual Basic y LabVIEW), así como la claridad de los controles y la estabilidad de la comunicación con el microcontrolador.
- Eficiencia energética: Se revisó el consumo eléctrico del sistema

durante la operación continua.

Los resultados de esta evaluación sirvieron para identificar oportunidades de mejora en el diseño y en la programación del sistema, así como para validar que el prototipo cumple su función como herramienta educativa para la enseñanza de robótica y automatización.

CONCLUSIONES

El desarrollo del brazo robótico didáctico de cinco grados de libertad cumplió satisfactoriamente con el objetivo general planteado al inicio del proyecto. Se logró diseñar, fabricar e integrar un prototipo funcional utilizando impresión 3D, servomotores y un microcontrolador, complementando con una interfaz de control desarrollada en entornos gráficos. El sistema demostró ser una herramienta útil para introducir a estudiantes a conceptos fundamentales de robótica y automatización, permitiendo visualizar de manera tangible la relación entre modelado teórico, diseño mecánico y control electrónico.

En el ámbito técnico, el proyecto permitió validar el proceso completo de construcción de un manipulador educativo, abarcando desde la inspección y adecuación del modelo inicial, la selección e integración de componentes electrónicos, el desarrollo del firmware y la interfaz de usuario, hasta la evaluación del desempeño del sistema. La implementación de la arquitectura del control, basada en Arduino nano, y servomotores, junto con la comunicación con LabView, evidencio la viabilidad de utilizar herramientas de bajo costo y acceso abierto para fines educativos.

Asimismo, el análisis teórico incorporado - que incluyo la formula la formulación de la ecuación de movimiento, la cinemática directa e inversa, y el estudio de los limites mecánicos – permitió establecer criterios de diseño y operaciones alineadas con los principios de la robótica manipuladora.

Desde la perspectiva educativa, el prototipo desarrollado aporta un recurso didáctico accesible y funcional, que facilita la comprensión de los

sistemas robóticos a través de la experimentación directa. La interacción con el brazo permite a los estudiantes relacionar el modelado matemático con la respuesta física del robot, fortaleciendo el aprendizaje significativo mediante el enfoque autodidacta.

Además, la interfaz gráfica diseñada contribuye a una curva de aprendizaje más suave, al permitir el control del robot sin requerir experiencia previa en programación textual. El desarrollo conllevo ciertas limitaciones al uso de componentes económicos y materiales impresos en 3D. Entre ellas se identifican la presencia de holgura mecánica en los servomotores, restricciones en la precisión de movimiento y una capacidad de carga limitada de PETG como material estructural. También se observaron limitaciones en el rendimiento del protocolo *Lifa Base* principalmente respecto a la velocidad de comunicación y a la independencia del puerto serial, lo cual restringe aplicaciones que requieren control más avanzado o en tiempo real.

Finalmente, se identifican áreas de mejor y áreas de trabajo futuro que pueden fortalecer el proyecto y amplían su impacto educativo.

Entre las principales se recomienda: integrar servomotores con retroalimentación para mejora la precisión y repetitibiliad, migrar hacia protocolos de comunicación más robustos o de menor latencia, incorporar sensores para implementar lazos de control cerrado y ampliar la interfaz gráfica con módulos de monitoreo, versión optimizada del modelo estructural, fabricada en materiales compuestos o con mecanismos de reducción de holgura podría incrementar el desempeño del brazo en escenarios más exigentes. Adicionalmente se sugiere desarrollar guías didácticas asociadas para facilitar su implementación en cursos de mecatrónica, robótica o automatización.

En síntesis, el proyecto alcanzo su propósito fundamental al demostrar que es posible desarrollar un brazo robótico de bajo costo funcional y con

respaldo teórico que sirva como herramienta formativa para la enseñanza de la robótica. El trabajo realizado sienta las bases para futuras iteraciones que puedan fortalecer tanto su rendimiento técnico como su contribución académica.

REFERENCIAS BIBLIOGRÁFICAS.

1. Tirado Robles, M. C. (2020). ¿Qué es un robot? Análisis jurídico comparado de las propuestas japonesas y europeas (No. ART-2020-118780).
2. Saquimux, C. B. (2005). Diseño y construcción de un brazo robótico. Universidad de San Carlos de Guatemala, Facultad de ingenierías, escuela de ingenierías en ciencia y sistema.
3. Colorado, R. M. (2016). Cinemática y dinámica de robots manipuladores. Alpha Editorial.
4. López Carrasquero, F. (2004). Fundamentos de polímeros. Escuela Venezolana para la enseñanza de la Química. Mérida, 49-51.
5. Toledo, I., Toledo, A., Vázquez, A., Santana, V., & Flores, A. (2013). Reciclaje y utilización de termoplásticos. In Ciencias de la Ingeniería y Tecnología Handbook T-III: Congreso Interdisciplinario de Cuerpos Académicos (pp. 171–181). Sitio web: https://dialnet. unirioja.

es/servlet/extart.

6. Romero, J. M. L. (2014). Transformación de materiales termoplásticos. QUIT0209. IC Editorial.
7. Jorquera Ortega, A. (2016). Fabricación digital: Introducción al modelado e impresión 3D. Ministerio de Educación, Cultura y Deporte.

Figura 66. Descripción de la imagen 66.

8. Noorani, R. (2017). 3D printing: technology, applications, and selection. CRC Press.
9. TCA Automation. (2024). ¿Qué es un servomotor y para qué sirve? Recuperado de https://www.tca-automation.com/servomotor-que-es-y-para-que-sirve/
10. Boletín UPIITA IPN. En aplicaciones mecatrónicas ¿Motor o Servomotor? Recuperado de https://www.boletin.upiita.ipn.mx/index.php/ciencia/577-cytnumero-44/1061-en-aplicaciones-mecatronicas-motor-o-servomotor
11. Mecatrónica LATAM. Servomotor. Recuperado de https://www.mecatronicalatam.com/es/tutoriales/motor/motores-electricos/motor-de-corriente-continua/servomotor/
12. Autsou, S., Kudelina, K., Vaimann, T., Rassõlkin, A., & Kallaste, A. (2024). Principles and methods of servomotor control: Comparative analysis and applications. Applied Sciences, 14(6), 2579.
13. Younkin, G. W. (2002). Industrial Servo Control Systems: Fundamentals And Applications, Revised And Expanded. CRC press.
14. Mijwel, M. M. (2018). What is Arduino Programming. no. March.
15. Decuir, F., Phelan, K., & Hollins, B. C. (2016, March). Mechanical strength of 3-D printed filaments. In 2016 32nd Southern Biomedical Engineering Conference (SBEC) (pp. 47-48). IEEE.

16. Martínez, C. A. H., & Rubio, J. P. F. (2014). Tecnologías de fabricación aditiva. La impresora 3D, antecedentes y funcionamiento. Ignis, (7), 24-30.
17. Nano, A. (2018). Arduino nano. A MOBICON Company, 30.
18. Creality. (s.f.). Wall. Creality Wiki. https://wiki.creality.com/en/software/update-released/Strength/wall
19. Creality. (s.f.). Parameter - Quality. Creality Wiki. https://wiki.creality.com/en/software/creality-print/parameter-quality
20. Creality Cloud. (2024, 30 de septiembre). Guía completa sobre el relleno en la impresión 3D. Creality Cloud. https://www.crealitycloud.com/es/blog/tutorials/infill-3d-printing
21. Organtini, G. (2015). Scientific Arduino Programming. Rome, Italy: Sapieza Univesità di Roma.
22. Budynas, R. G., & Nisbett, J. K. (2011). Shigley's mechanical engineering design (Vol. 9, pp. 409-473). New York: McGraw-Hill.
23. E. Chávez Mendiola y J. O. Rivera Nieblas, *Software e interfaces de control de servomotores del tipo brushless, con aplicación en robot paralelo*, Tesis de Maestría, Instituto Tecnológico Superior de Cajeme, Hermosillo, Sonora, México, 2008. [En línea]. Disponible en: https://www.researchgate.net/publication/324248781_Software_e_interfaces_de_control_de_servomotores_del_tipo_brushless_con_aplicacion_en_robot_paralelo
24. A. Barrientos, L. F. Peñín, C. Balaguer, y R. Aracil, *Fundamentos de robótica*, 2ª ed. Madrid, España: McGraw-Hill Interamericana de España, 2007.
25. J. D. Trujillo, D. Perdomo Trujillo, J. G. Gómez, J. G. Vera, y M. R. Arbulú, «Ortesis de Mano Robótica», #AS, vol. 1, n.º 20, pp. 30-48, ago. 2022.
26. Steren, Micro servomotor con torque de 2,2 Kgf/cm (MOT-110), Steren Electrónica. [En línea]. Disponible en:

https://www.steren.com.mx/micro-servomotor-con-torque-de-2-2-kgf-cm.html [Accedido: 2-jun-2025].

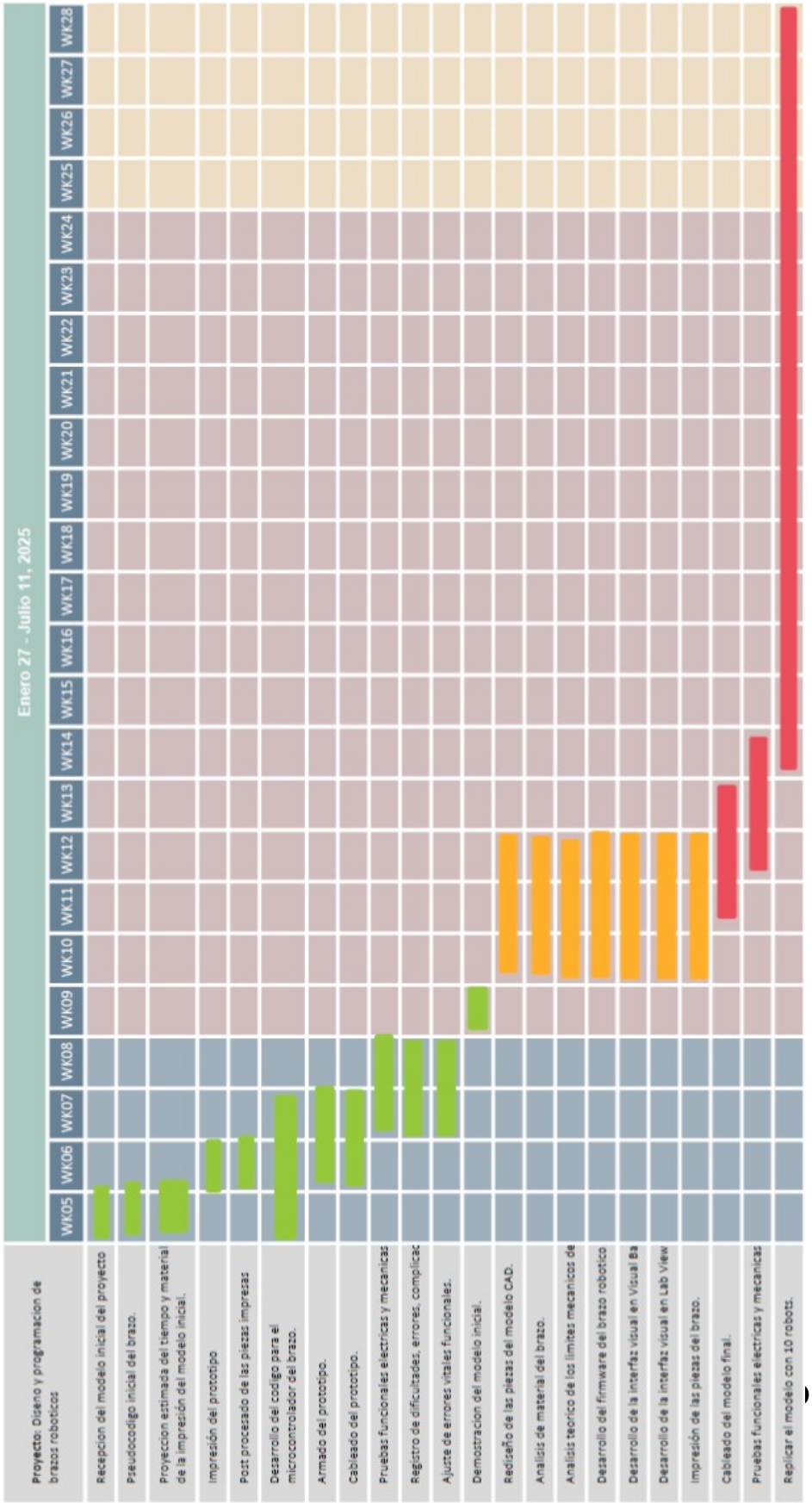
27. TowerPro, *MG995 High Speed Metal Gear Dual Ball Bearing Servo*, Datasheet, Components101. [En línea]. Disponible en: https://components101.com/sites/default/files/component_datasheet/MG995-Servo-Motor-Datasheet.pdf [Accedido: 2-jun-2025]

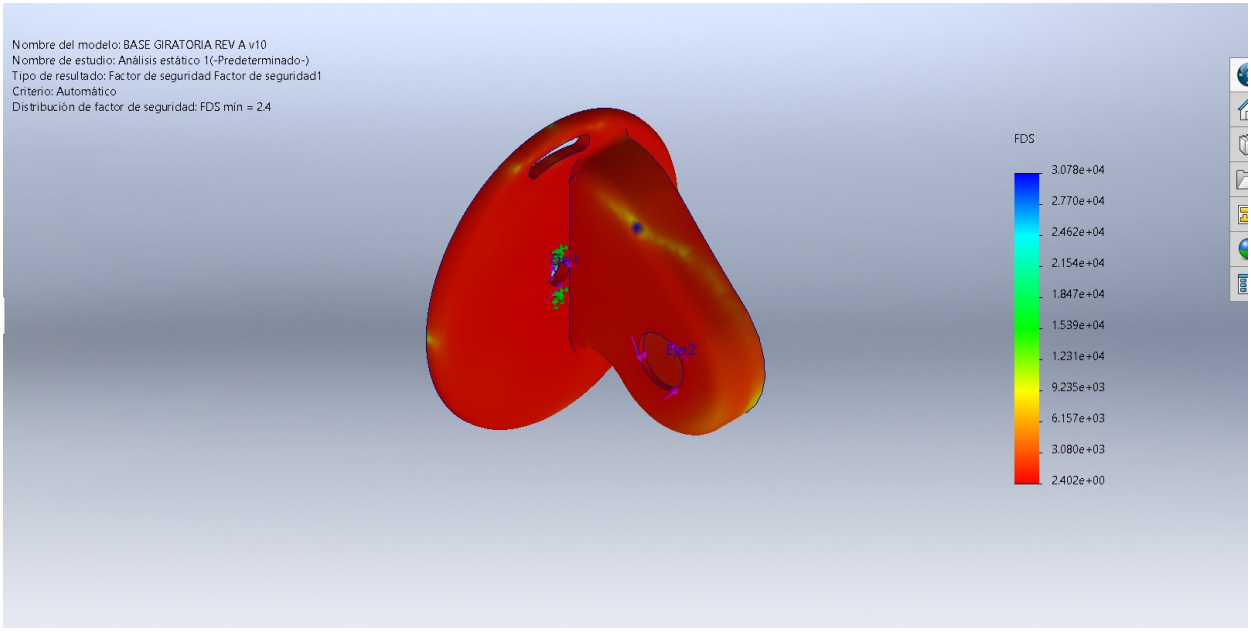
28. M. Molina Cárdenas, P. Pedroza Barrios, K. M. Gaitán Moreno, J. F. Salgado Arismendy y M. C. Ordóñez Ávila, "Diseño y Construcción del Prototipo de un Brazo Robótico con Tres Grados de Libertad, como Objeto de Estudio," *Ingeniare*, no. 18, pp. 87–94, 2015. [En línea]. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=5478783. [Accedido: 3-jun-2025].

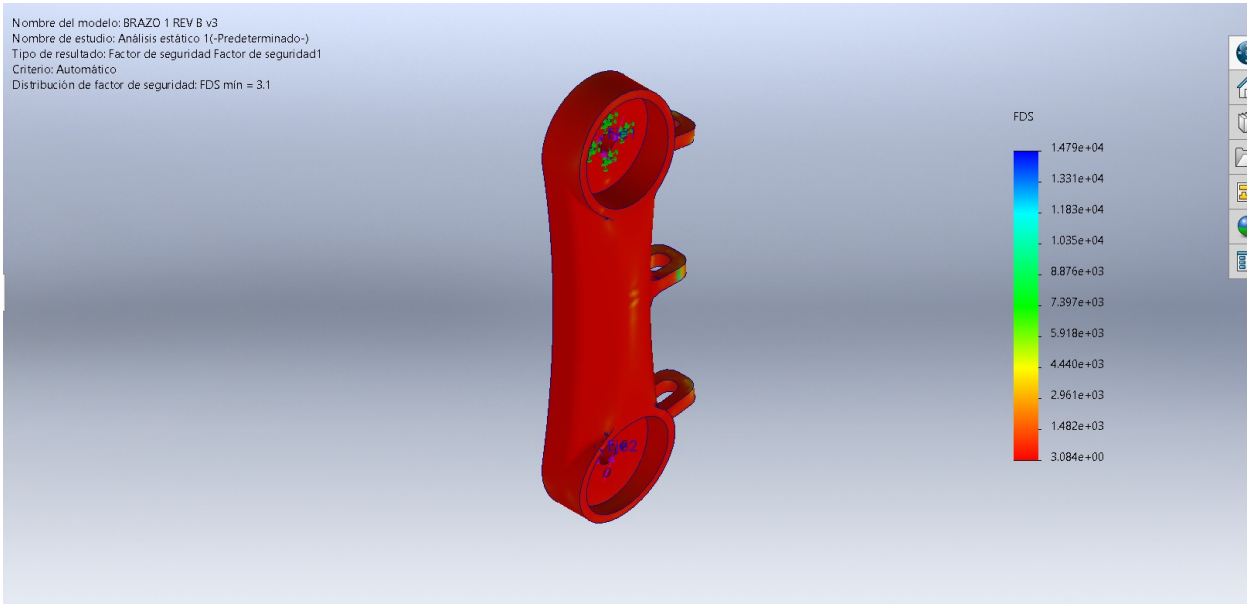
ANEXOS

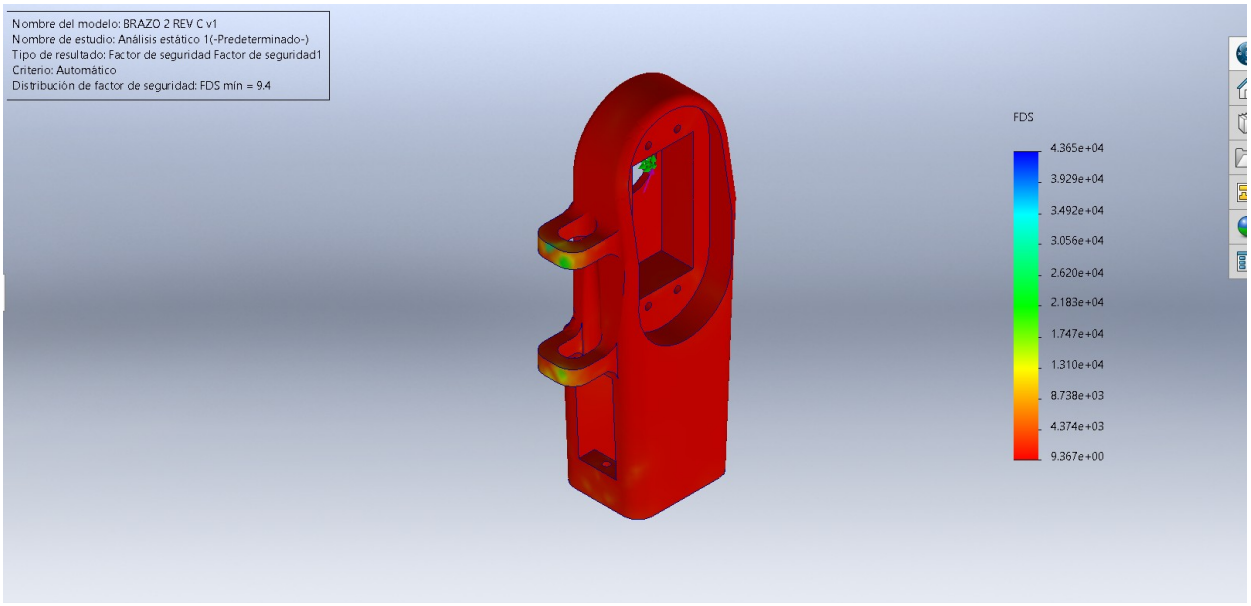
ANEXO 1. CRONOGRAMA PRELIMINAR DE ACTIVIDADES

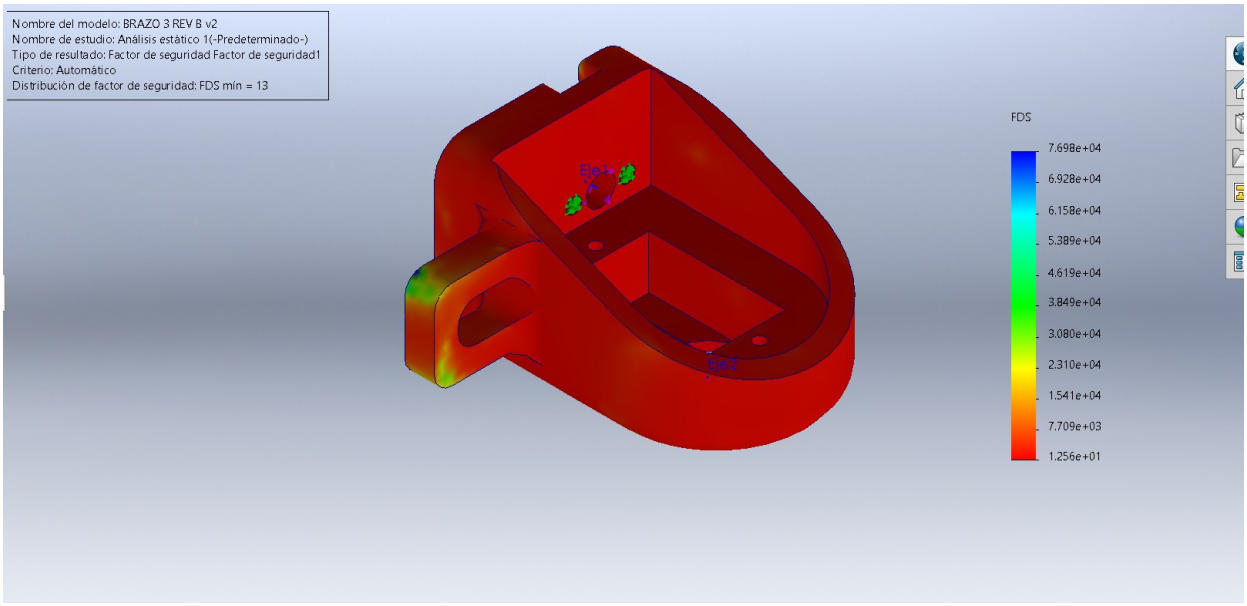
ANEXO **OLIDWORKS**
(ESCAL

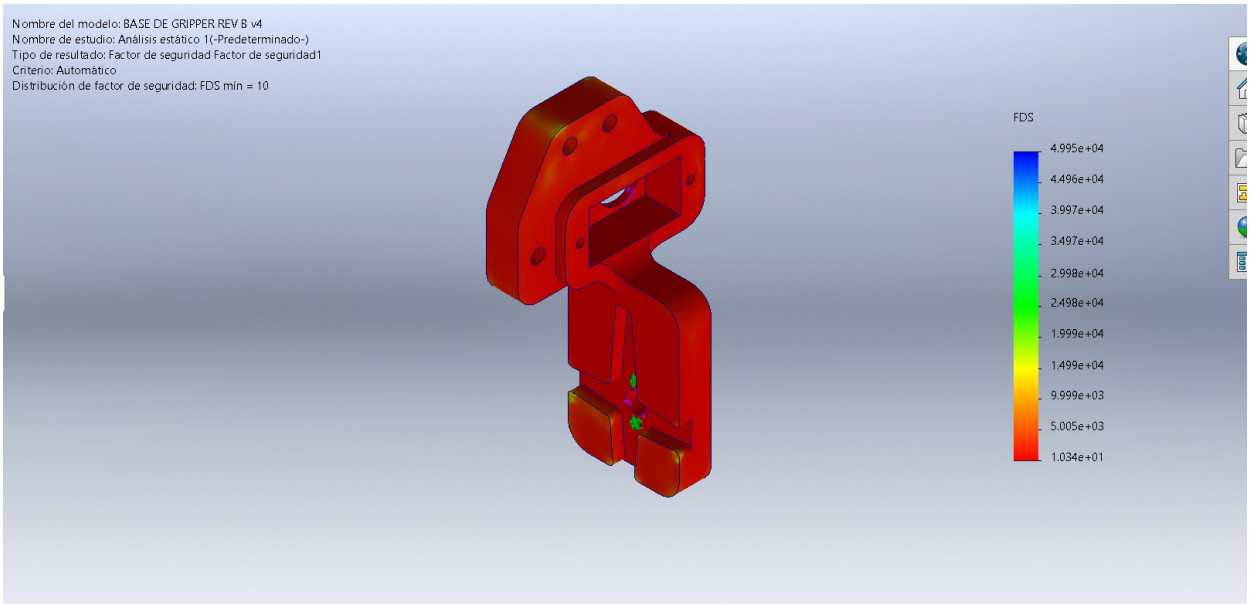


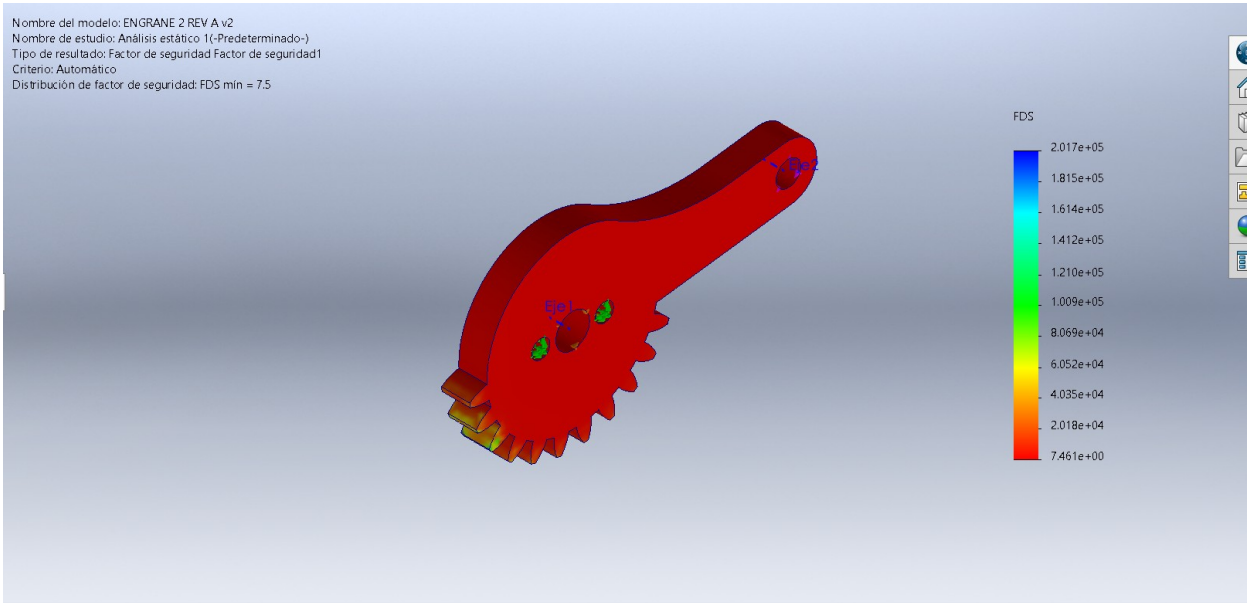

Nombre del modelo: BASE GIRATORIA REV A v10
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 2.4
FDS
3.078e+04
2.770e+04
2.462e+04
2.154e+04
1.847e+04
1.539e+04
1.231e+04
9.235e+03
6.157e+03
3.080e+03
2.402e+00

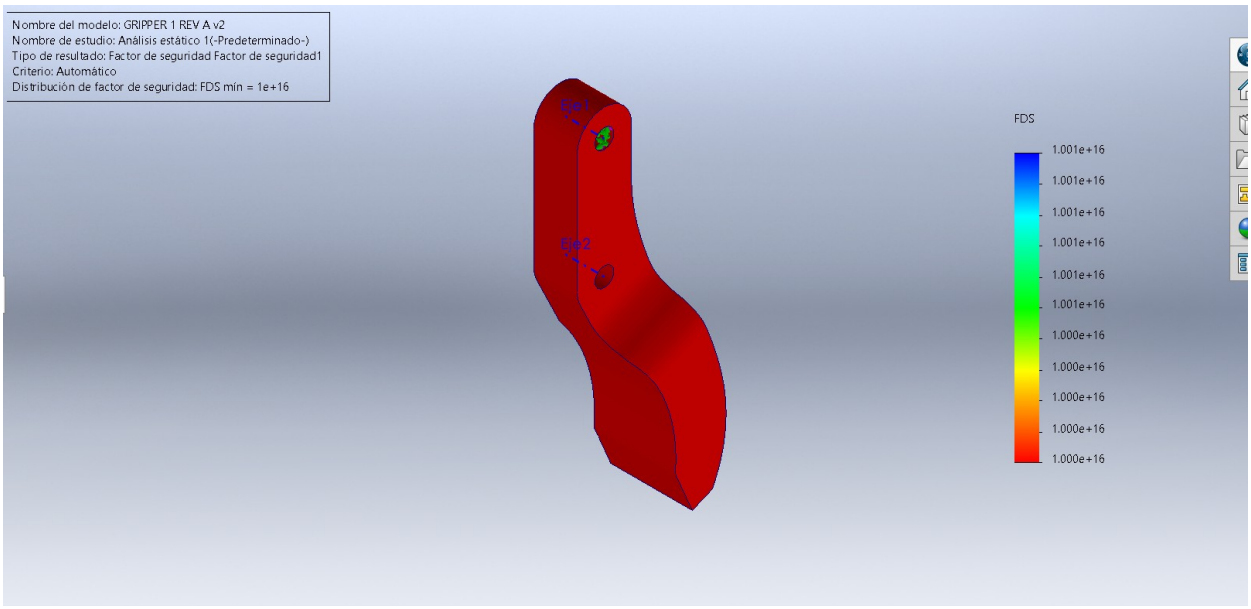

Nombre del modelo: BRAZO 1 REV B v3
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 3.1
FDS
1.479e+04
1.331e+04
1.183e+04
1.035e+04
8.876e+03
7.397e+03
5.918e+03
4.440e+03
2.961e+03
1.482e+03
3.084e+00

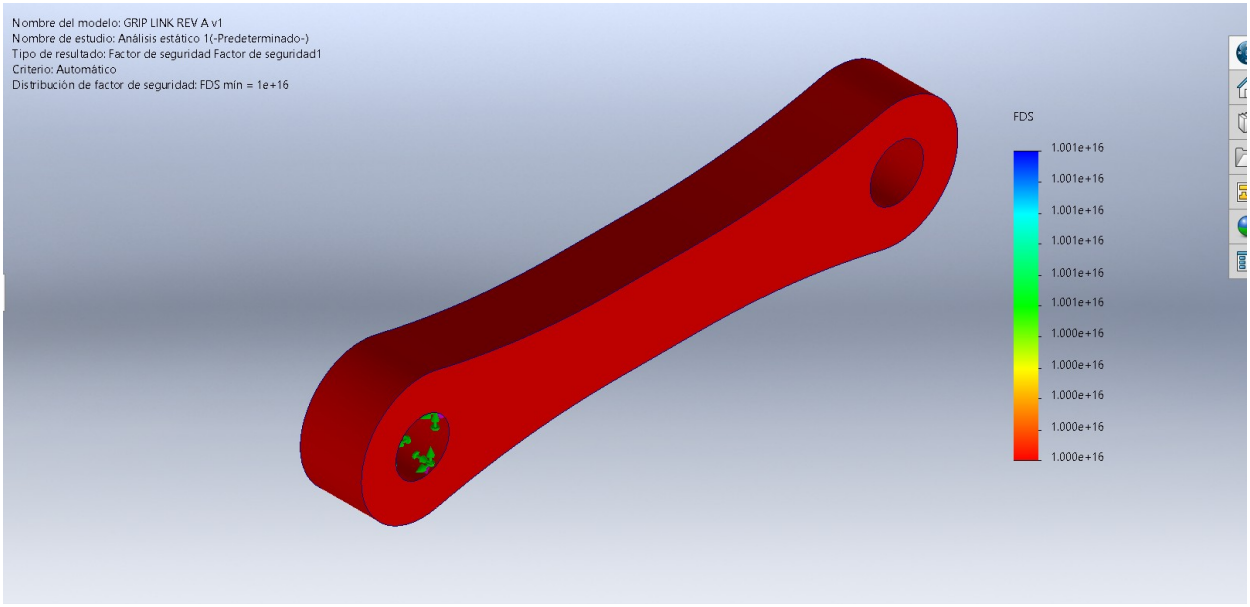

Nombre del modelo: BRAZO 2 REV C v1
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 9.4
FDS
4.365e+04
3.929e+04
3.492e+04
3.056e+04
2.620e+04
2.183e+04
1.747e+04
1.310e+04
8.738e+03
4.374e+03
9.367e+00


Nombre del modelo: BRAZO 3 REV B v2
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 13
FDS
7.698e+04
6.928e+04
6.158e+04
5.389e+04
4.619e+04
3.849e+04
3.080e+04
2.310e+04
1.541e+04
7.709e+03
1.256e+01

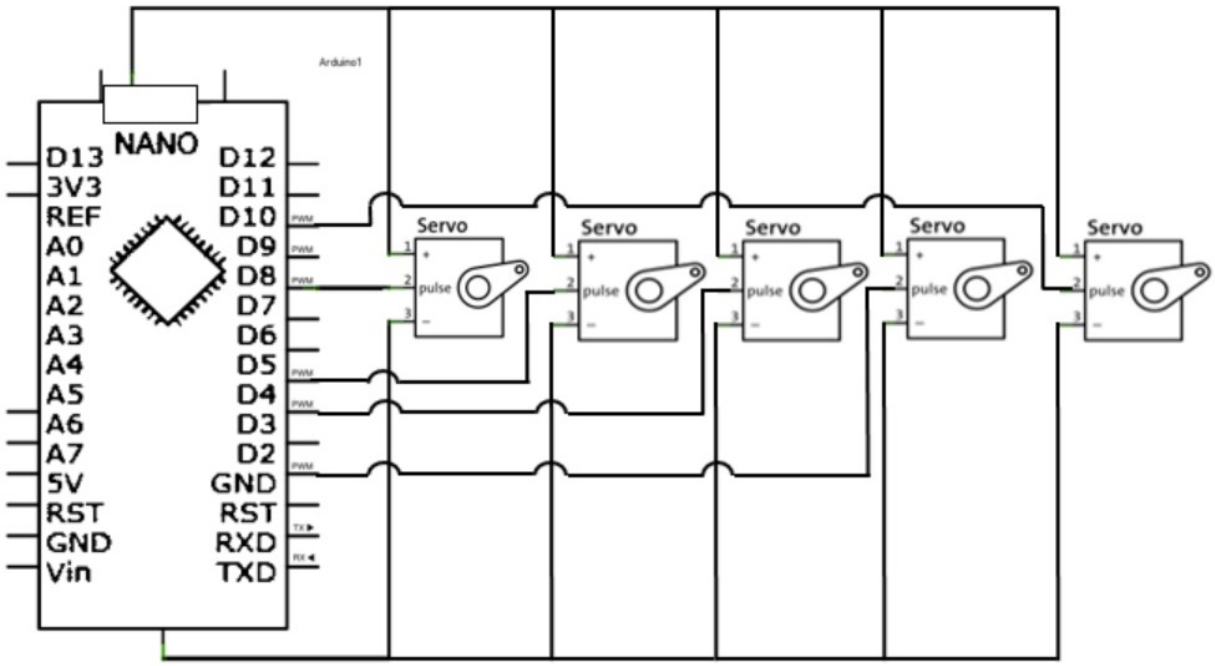

Nombre del modelo: BASE DE GRIPPER REV B v4
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 10
FDS
4.995e+04
4.496e+04
3.997e+04
3.497e+04
2.998e+04
2.498e+04
1.999e+04
1.499e+04
9.999e+03
5.005e+03
1.034e+01


Nombre del modelo: ENGRANE 2 REV A v2
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 7.5
FDS
2.017e+05
1.815e+05
1.614e+05
1.412e+05
1.210e+05
1.009e+05
8.069e+04
6.052e+04
4.035e+04
2.018e+04
7.461e+00


Nombre del modelo: GRIPPER 1 REV A v2
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 1e+16
Eje1
Eje2
FDS
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.000e+16
1.000e+16
1.000e+16
1.000e+16
1.000e+16


Nombre del modelo: GRIP LINK REV A v1
Nombre de estudio: Análisis estático 1(-Predeterminado-)
Tipo de resultado: Factor de seguridad Factor de seguridad1
Criterio: Automático
Distribución de factor de seguridad: FDS mín = 1e+16
FDS
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.001e+16
1.000e+16
1.000e+16
1.000e+16
1.000e+16
1.000e+16

ANEXO 3. DIAGRAMA ELECTRÓNICO DE CONEXIONES.



ANEXO 4. CÓDIGO DE FIRMWARE DE ARDUINO: Lifa Base para Lab View

```
#include <Servo.h>

// Crear objetos para los servos
Servo q1;
Servo q2;
Servo q3;
Servo q4;
Servo gripper; // Servo adicional para el gripper

// Definir los pines donde están conectados los servos
int pin_q1 = 11; // Cambia este valor según tu configuración
int pin_q2 = 10;
int pin_q3 = 9;
int pin_q4 = 6;
int pin_gripper = 5; // Cambia este valor según tu configuración

void setup() {
  Serial.begin(115200);

  // Iniciar los servos en los pines correspondientes
  q1.attach(pin_q1);
  q2.attach(pin_q2);
  q3.attach(pin_q3);
  q4.attach(pin_q4);
  gripper.attach(pin_gripper);

  // Enviar valores iniciales
  Serial.println("0,0,90,0");
```

Guardado.

Residencias

```
  // Ajustar los servos a los valores iniciales
  q1.write(map(0, -90, 90, 180, 0));
  q2.write(map(0, -90, 90, 180, 0));
  q3.write(map(90, -90, 90, 180, 0));
  q4.write(map(0, -90, 90, 180, 0));

  Serial.println("Ingrese los ángulos para cada servo (q1, q2, q3, q4) separados por comas:");
  Serial.println("Escriba 'L' para cerrar el gripper (0°) o 'K' para abrirlo (90°):");
}

void loop() {
  if (Serial.available() > 0) {
    // Leer los valores ingresados
    String input = Serial.readStringUntil('\n');
    input.trim(); // Eliminar posibles espacios en blanco

    if (input == "L") {
      // Mover el gripper a 0 grados
      gripper.write(15);
      Serial.println("Gripper cerrado (0°)");
    } else if (input == "K") {
      // Mover el gripper a 90 grados
      gripper.write(70);
      Serial.println("Gripper abierto (90°)");
    } else {
      // Parsear la entrada para obtener los valores de los ángulos
      int q1_angle, q2_angle, q3_angle, q4_angle;
      if (sscanf(input.c_str(), "%d,%d,%d,%d", &q1_angle, &q2_angle, &q3_angle, &q4_angle) == 4) {
```

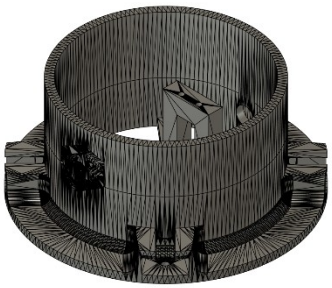
Guardado.

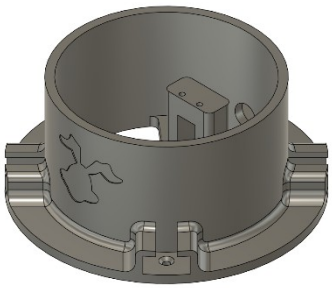
Residencias

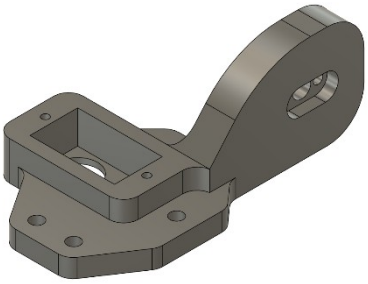
```
    int q1_angle, q2_angle, q3_angle, q4_angle;
    if (sscanf(input.c_str(), "%d,%d,%d,%d", &q1_angle, &q2_angle, &q3_angle, &q4_angle) == 4) {
      // Mapear los valores ingresados a los rangos de los servos
      int servo1_angle = map(q1_angle, -90, 90, 180, 0);
      int servo2_angle = map(q2_angle, -90, 90, 180, 0);
      int servo3_angle = map(q3_angle, -90, 90, 180, 0);
      int servo4_angle = map(q4_angle, -90, 90, 180, 0);

      // Ajustar los servos a los nuevos ángulos
      q1.write(servo1_angle);
      q2.write(servo2_angle);
      q3.write(servo3_angle);
      q4.write(servo4_angle);

      // Imprimir los ángulos para verificación
      Serial.print("Servo 1: ");
      Serial.print(servo1_angle);
      Serial.print(" | Servo 2: ");
      Serial.print(servo2_angle);
      Serial.print(" | Servo 3: ");
      Serial.print(servo3_angle);
      Serial.print(" | Servo 4: ");
      Serial.println(servo4_angle);

      Serial.println("Ingrese los nuevos ángulos para cada servo:");
    } else {
      Serial.println("Entrada inválida. Asegúrese de ingresar cuatro valores separados por comas o un comando válido ('L' o 'K').");
    }
  }
```

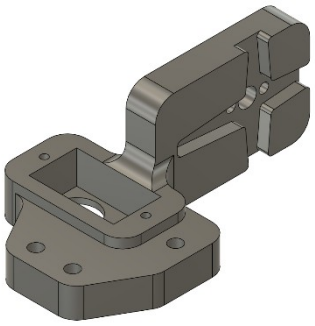
Guardado.

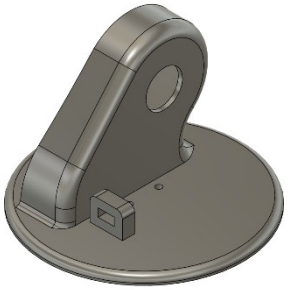
ANEXO 5. REDISEÑOS POR PIEZA

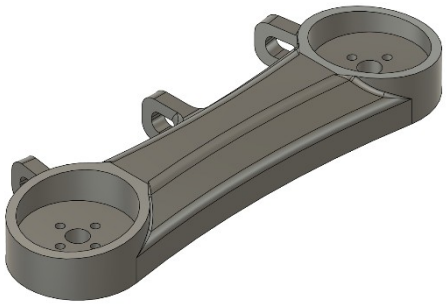


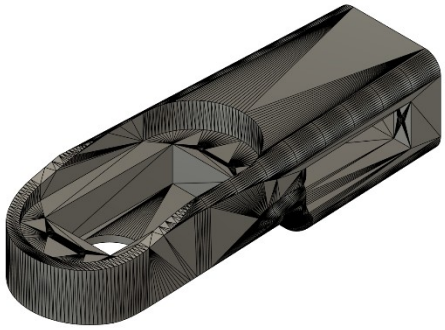


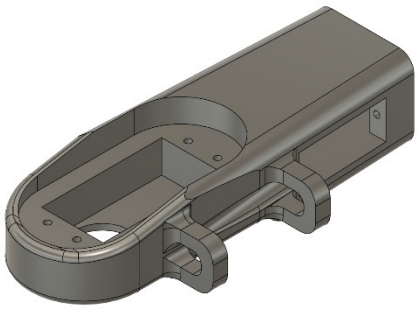


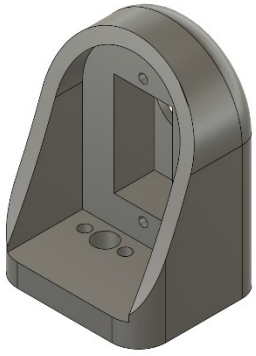


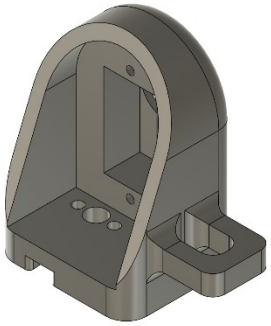


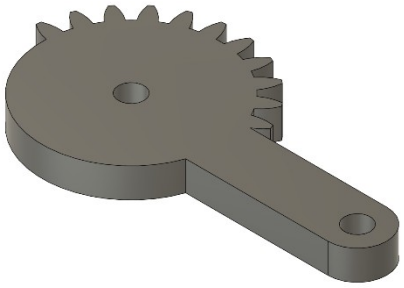


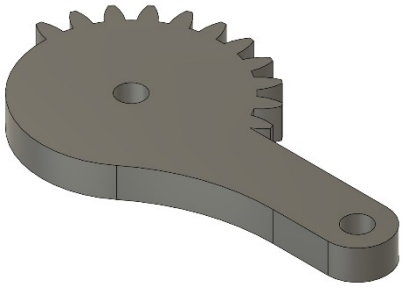


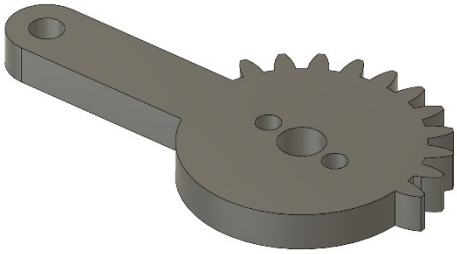


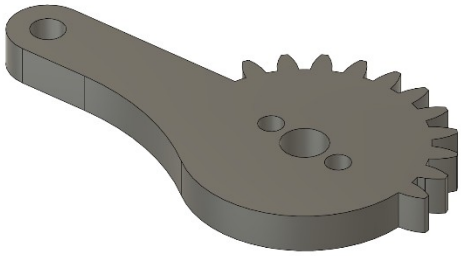


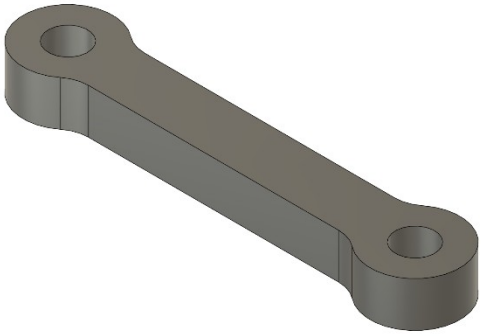


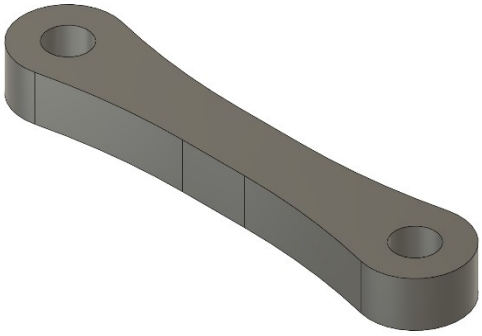








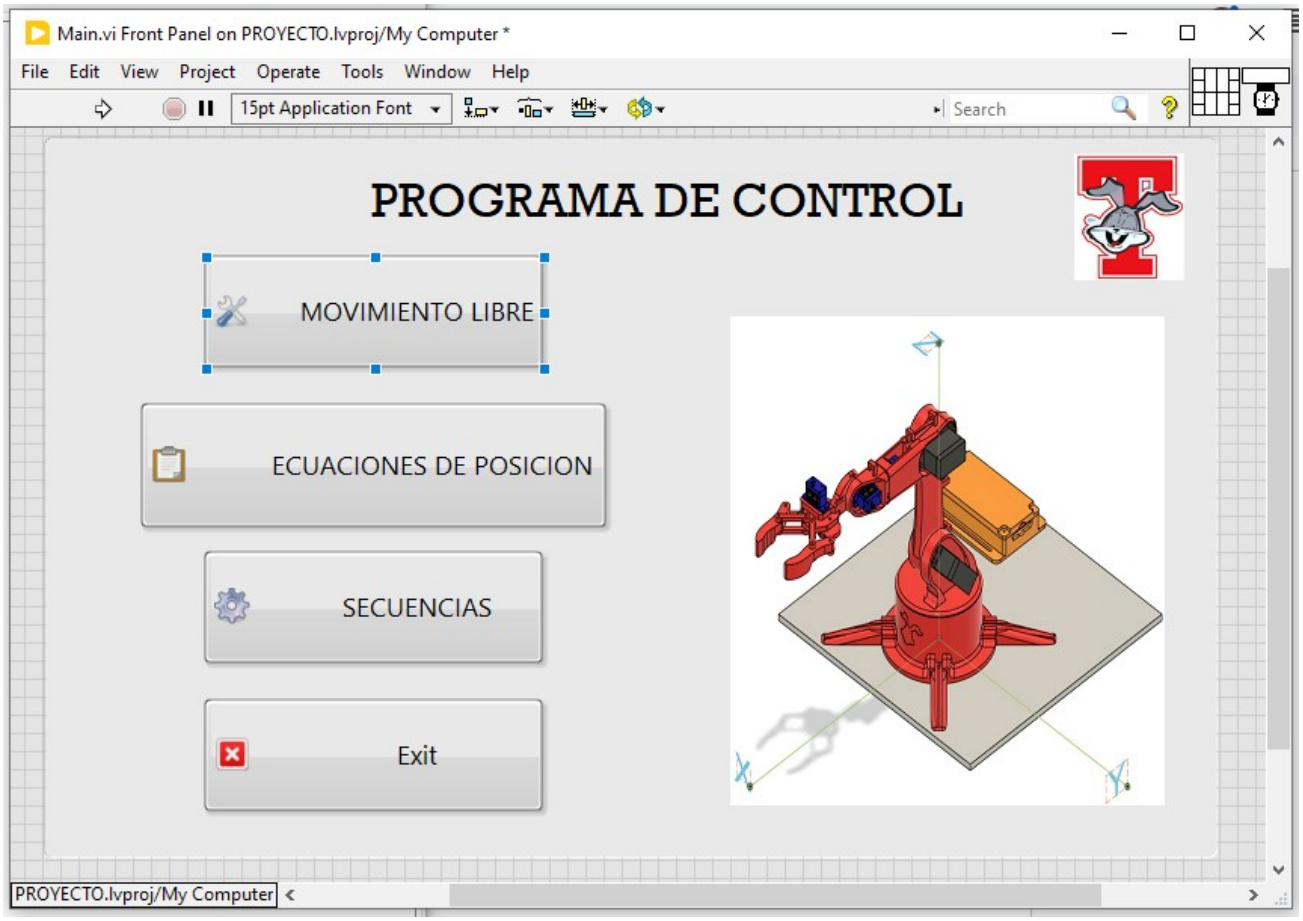


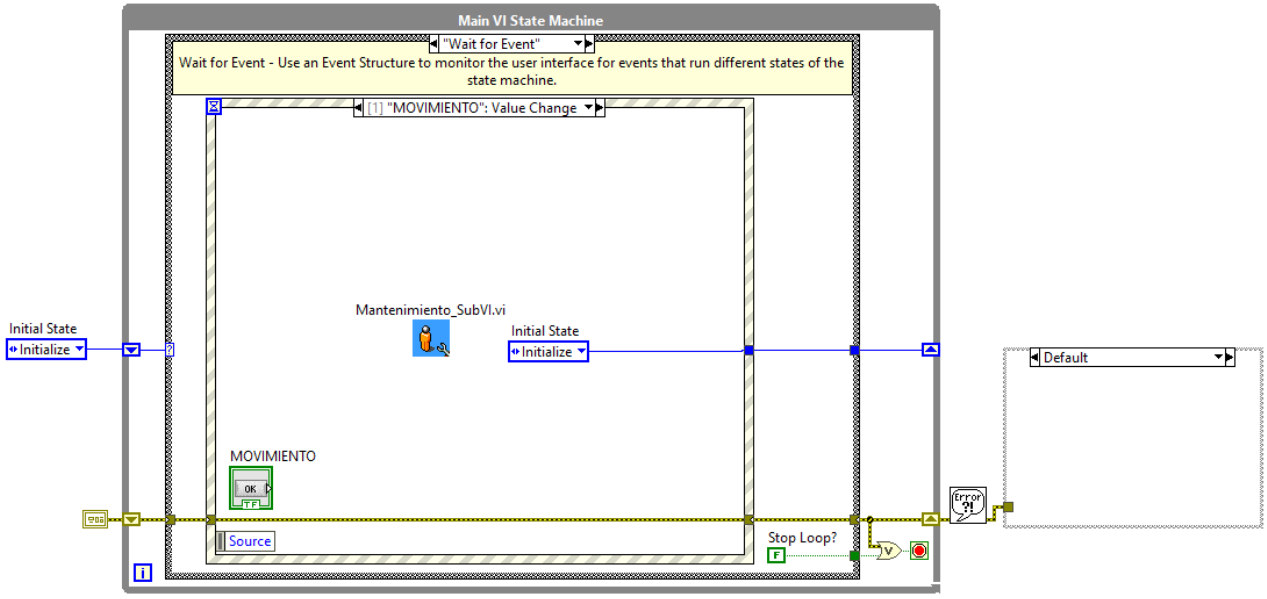


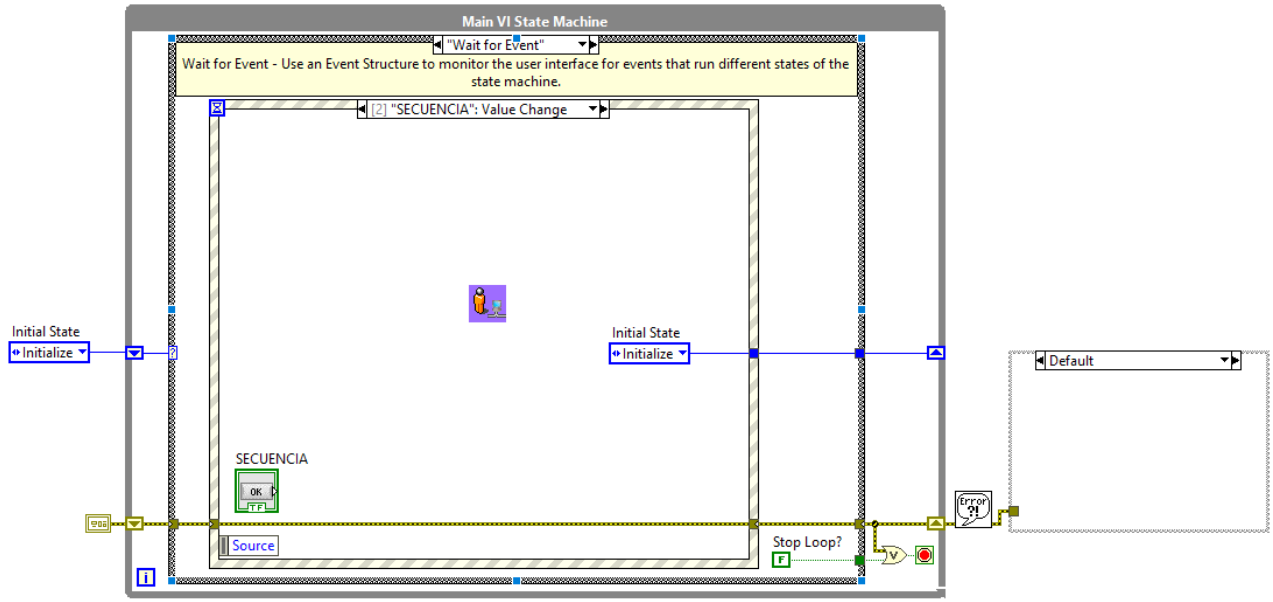
ANEXO 6. LISTA DE MATERIALES (BOM)

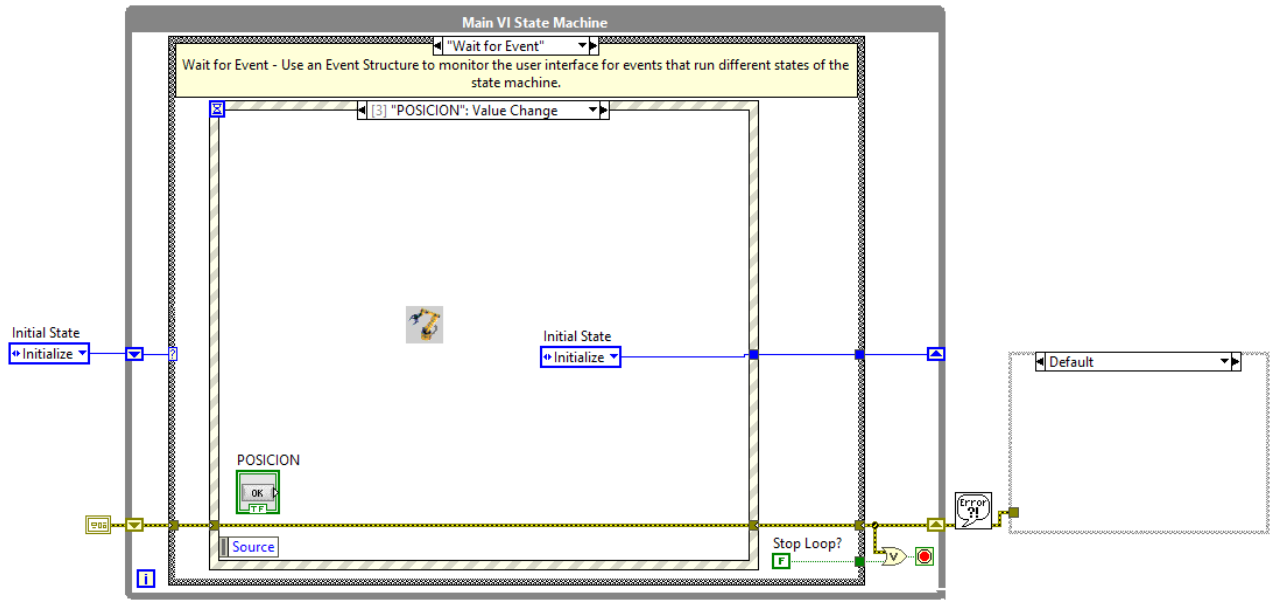
Tablilla	Perforada	Genérica	Electronica	1	$12,00	$12,00
Soldadura	1 rollo, 17 gramos	Genérica	Electronica	1	$60,00	$60,00
Cables	5 rollos, 10 metros	Genérica	Electronica	1	$260,00	$260,00
MDF bases	9 mm x 1.22 m x 2.40 m	Generico	Hardware	1	$70,00	$70,00
Tornillo M3	Cruz, 1 cm largo	Generico	Tornilleria	24	2,5	$60,00
Tornillo 5-40	1" largo	Generico	Tornilleria	7	$3,00	$21,00
Tornillo 5-40	1/2" largo	Generico	Tornilleria	3	$3,00	$9,00
					Total	$3.320,00

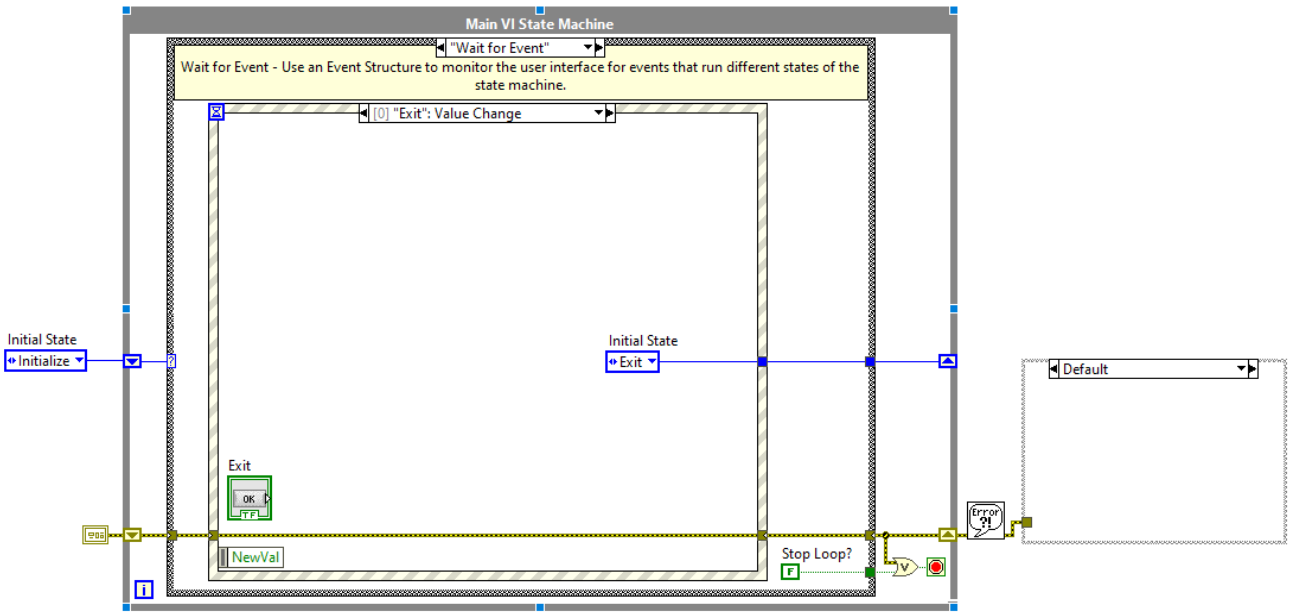
ANEXO 7. CAPTURAS DEL PROGRAMA EN LABVIEW

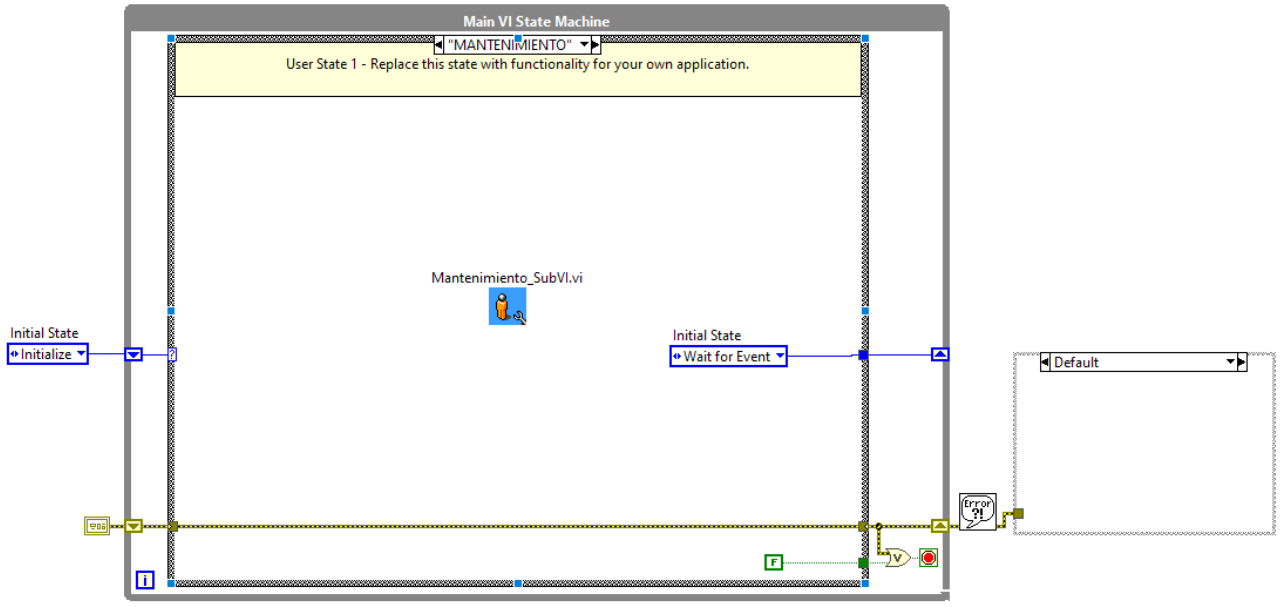


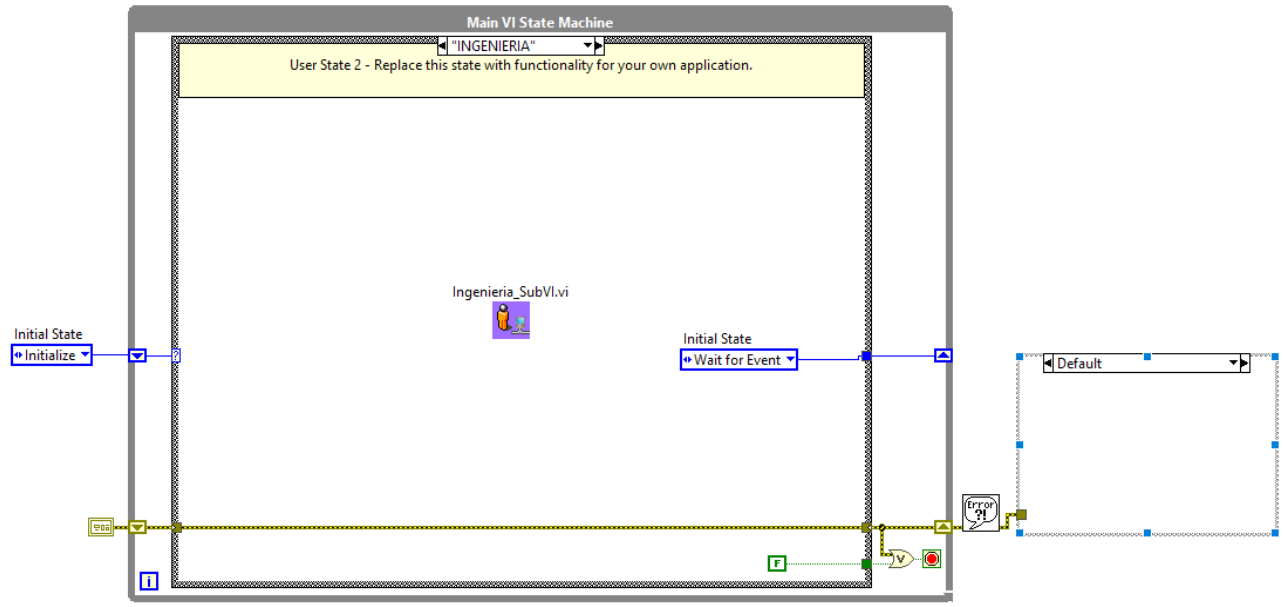


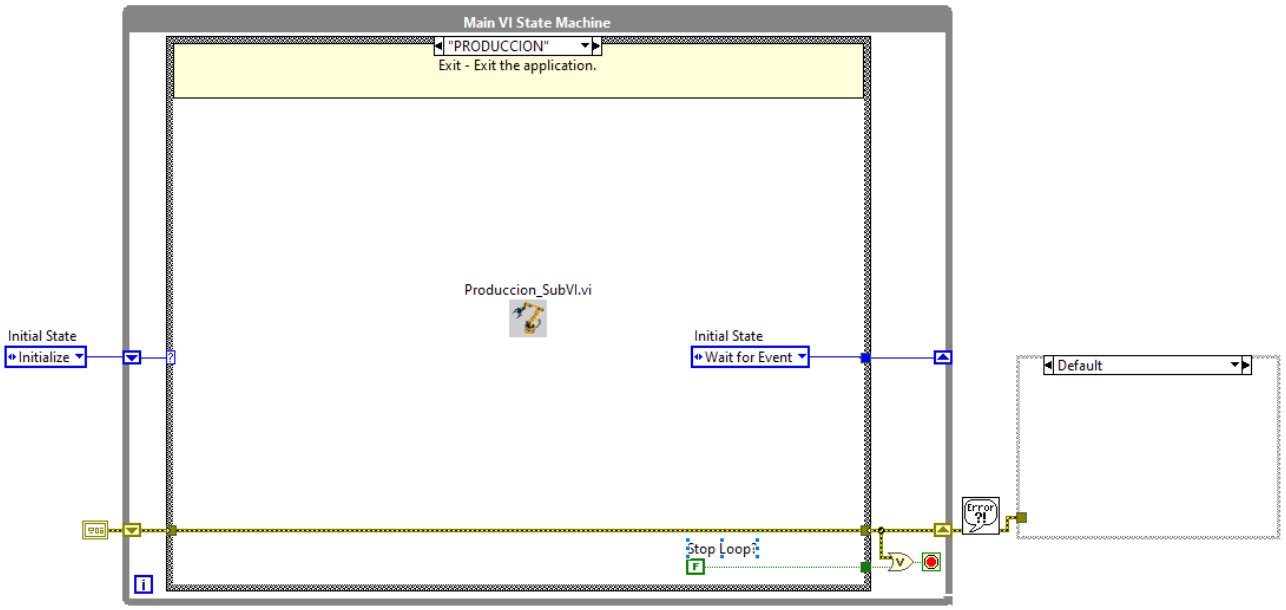

Main VI State Machine
"Wait for Event"
Wait for Event - Use an Event Structure to monitor the user interface for events that run different states of the state machine.
[2] "SECUENCIA": Value Change
Initial State
Initialize
Initial State
Initialize
Default
SECUENCIA
OK
TF
Source
Stop Loop?
Error

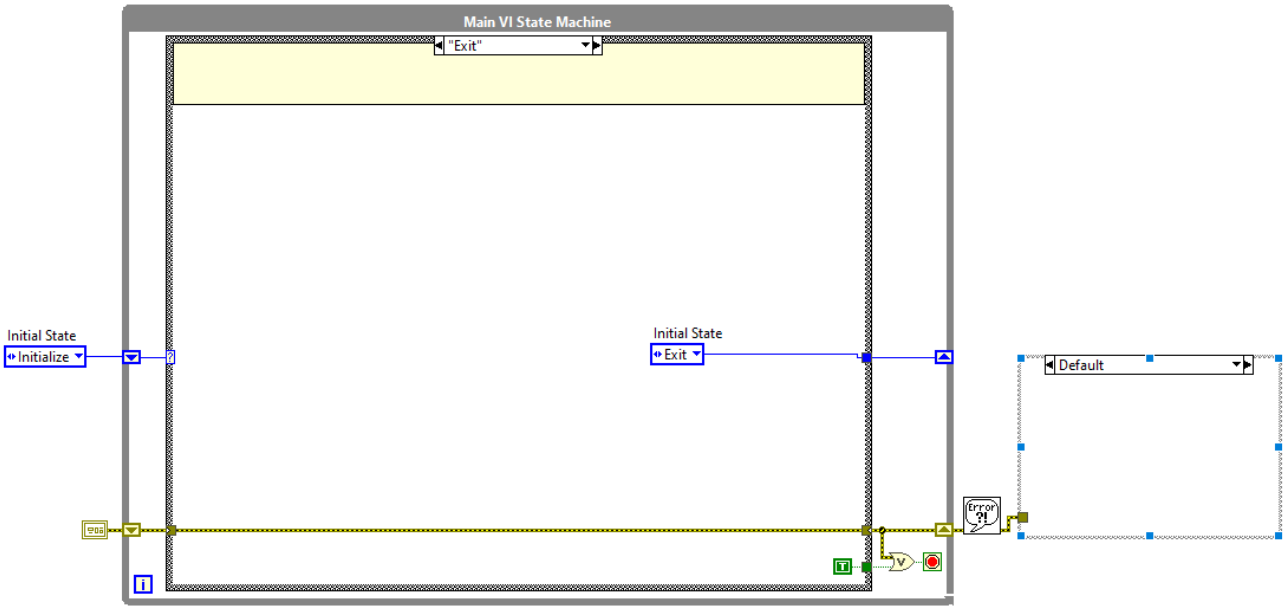

Main VI State Machine
"Wait for Event"
Wait for Event - Use an Event Structure to monitor the user interface for events that run different states of the state machine.
[3] "POSICION": Value Change
Initial State
Initialize
Initial State
Initialize
Default
POSICION
OK
TF
Source
Stop Loop?
Error

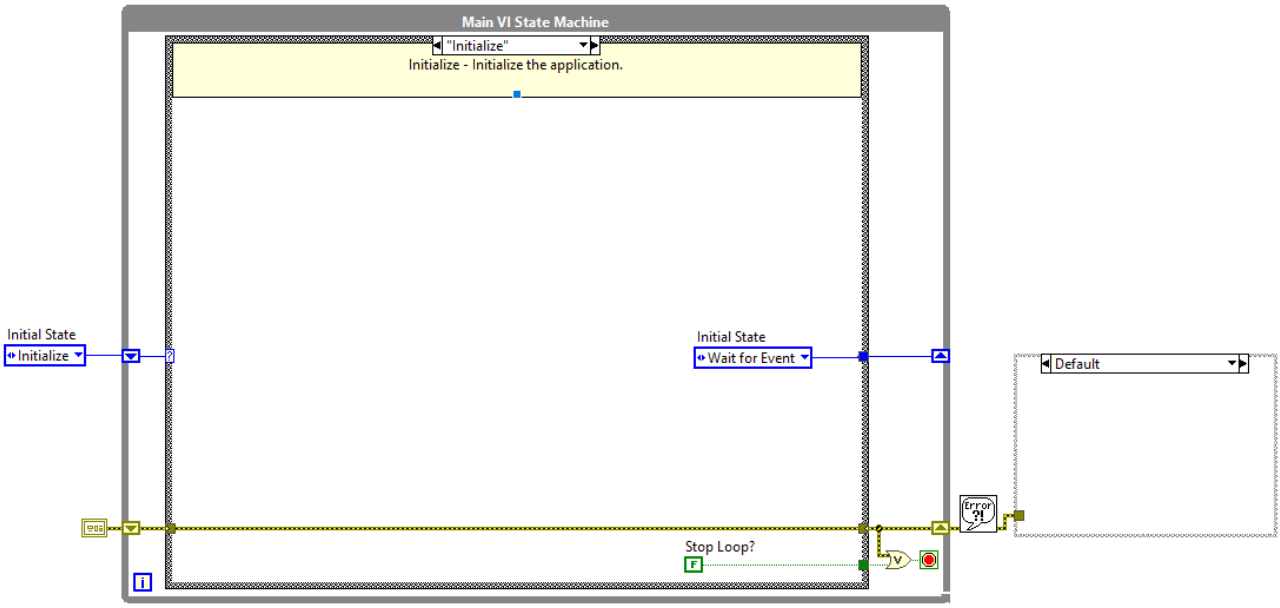

Main VI State Machine
"Wait for Event"
Wait for Event - Use an Event Structure to monitor the user interface for events that run different states of the state machine.
[0] "Exit": Value Change
Initial State
Initialize
Initial State
Exit
Default
Exit
OK
TF
NewVal
Stop Loop?
F
TF
Error

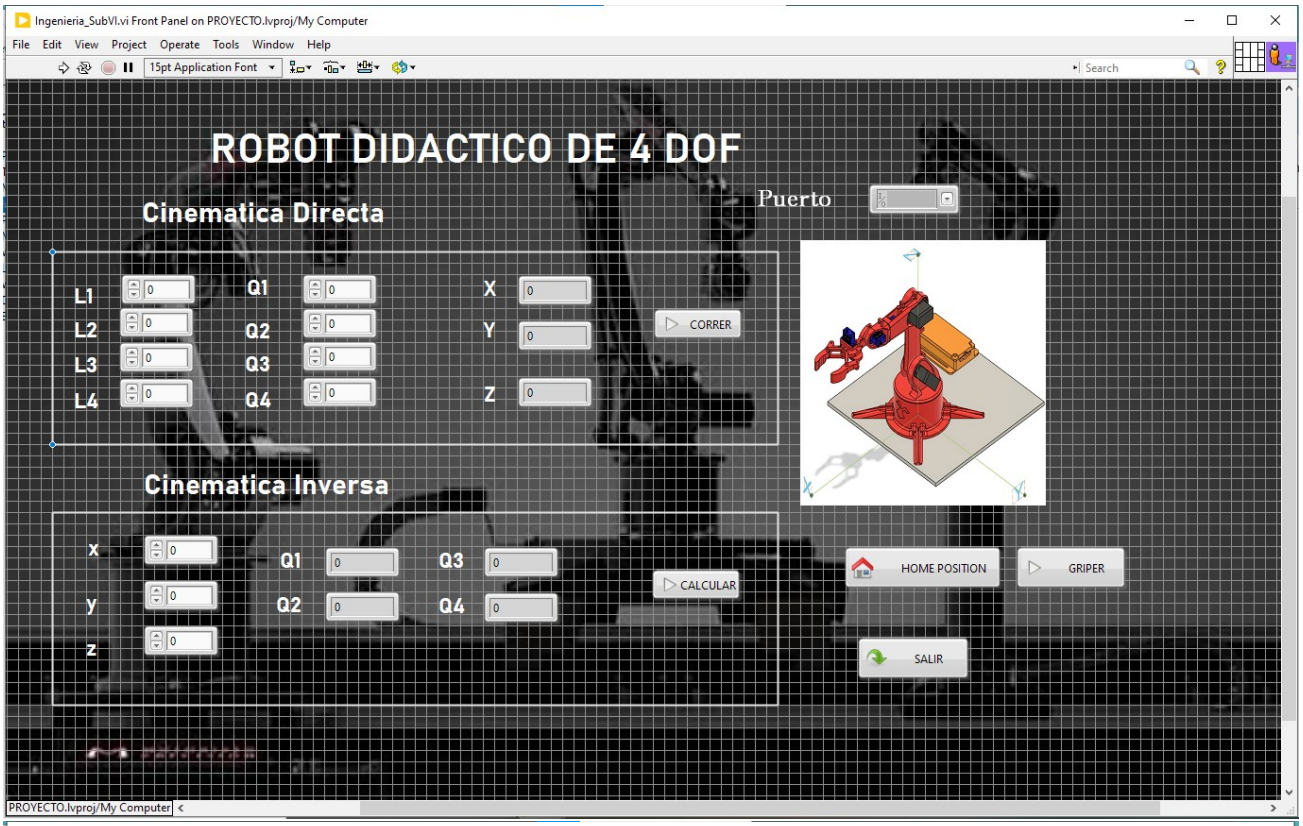

Main VI State Machine
"MANTENIMIENTO"
User State 1 - Replace this state with functionality for your own application.
Mantenimiento_SubVI.vi
Initial State
Initialize
Initial State
Wait for Event
Default
F
TF
Error

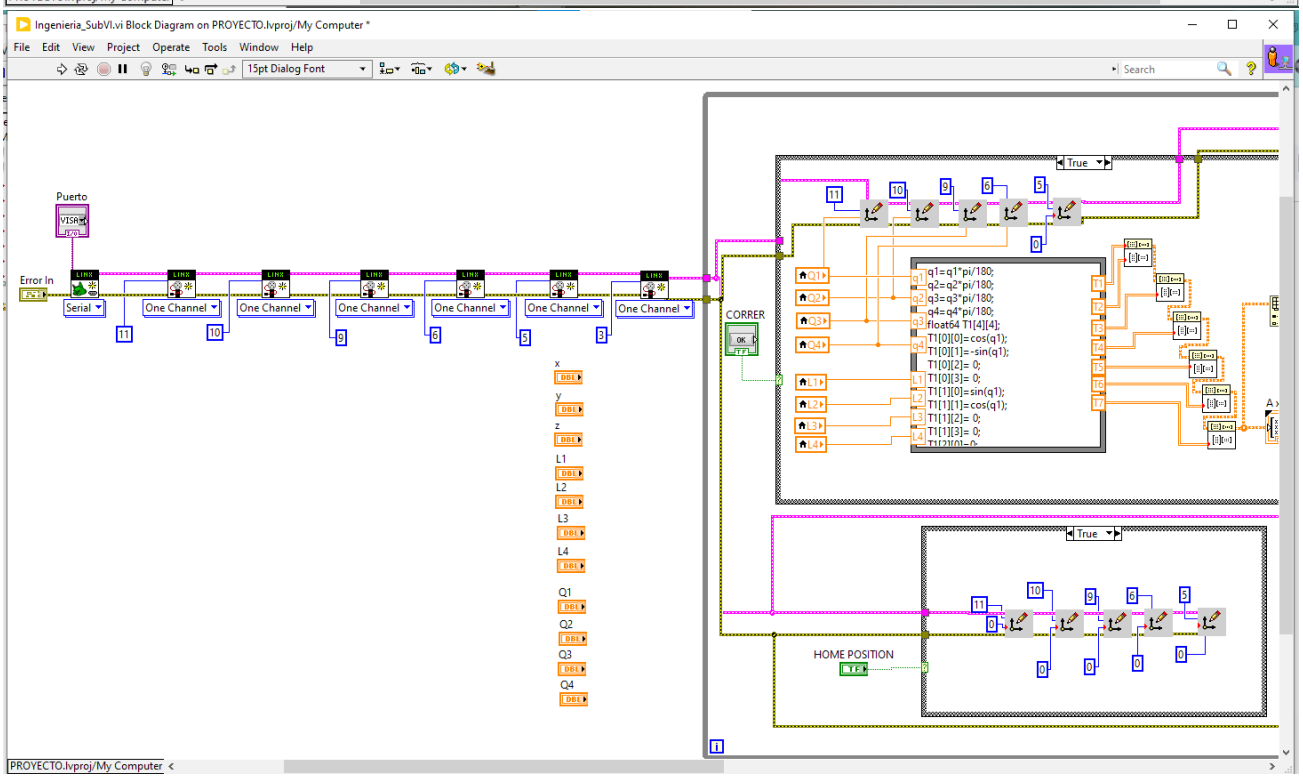

Main VI State Machine
"INGENIERIA"
User State 2 - Replace this state with functionality for your own application.
Ingenieria_SubVI.vi
Initial State
Initialize
Initial State
Wait for Event
Default
Error

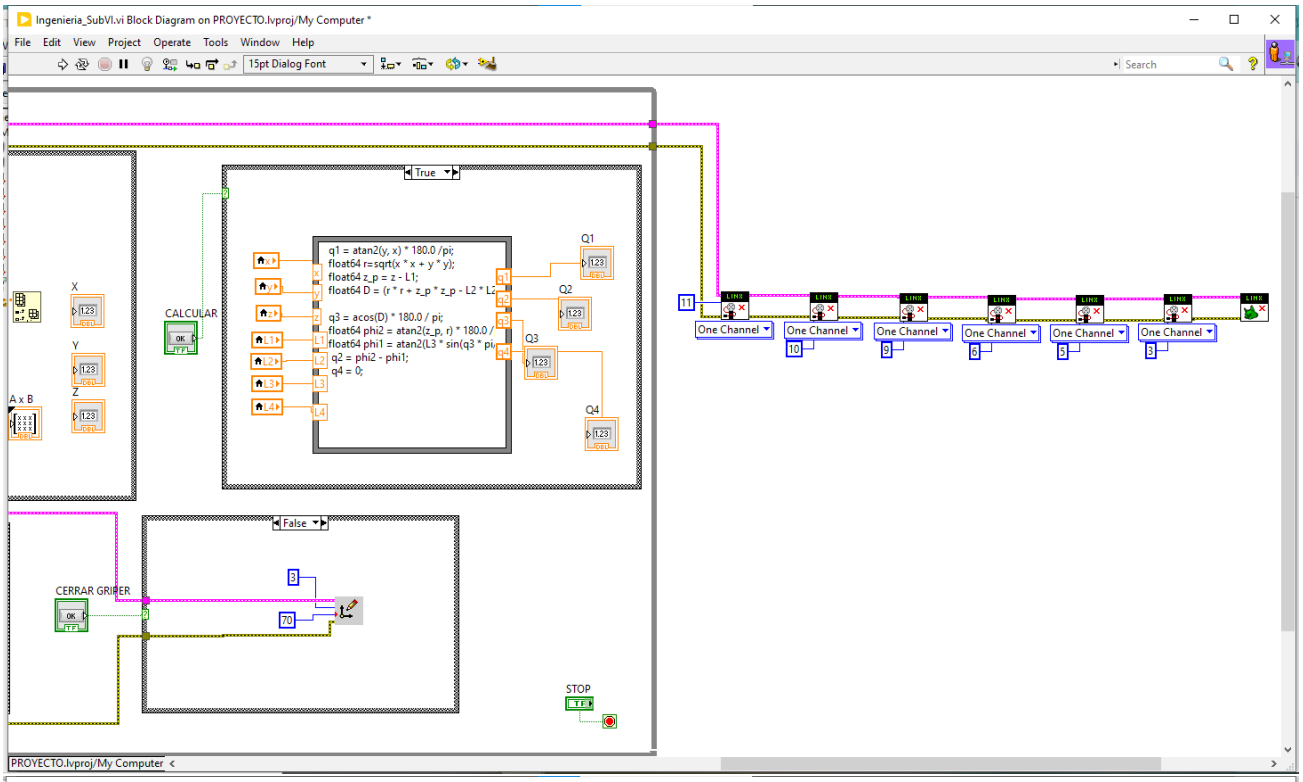

Main VI State Machine
"PRODUCCION"
Exit - Exit the application.
Produccion_SubVI.vi
Initial State
Initialize
Initial State
Wait for Event
Default
Error
Stop Loop?

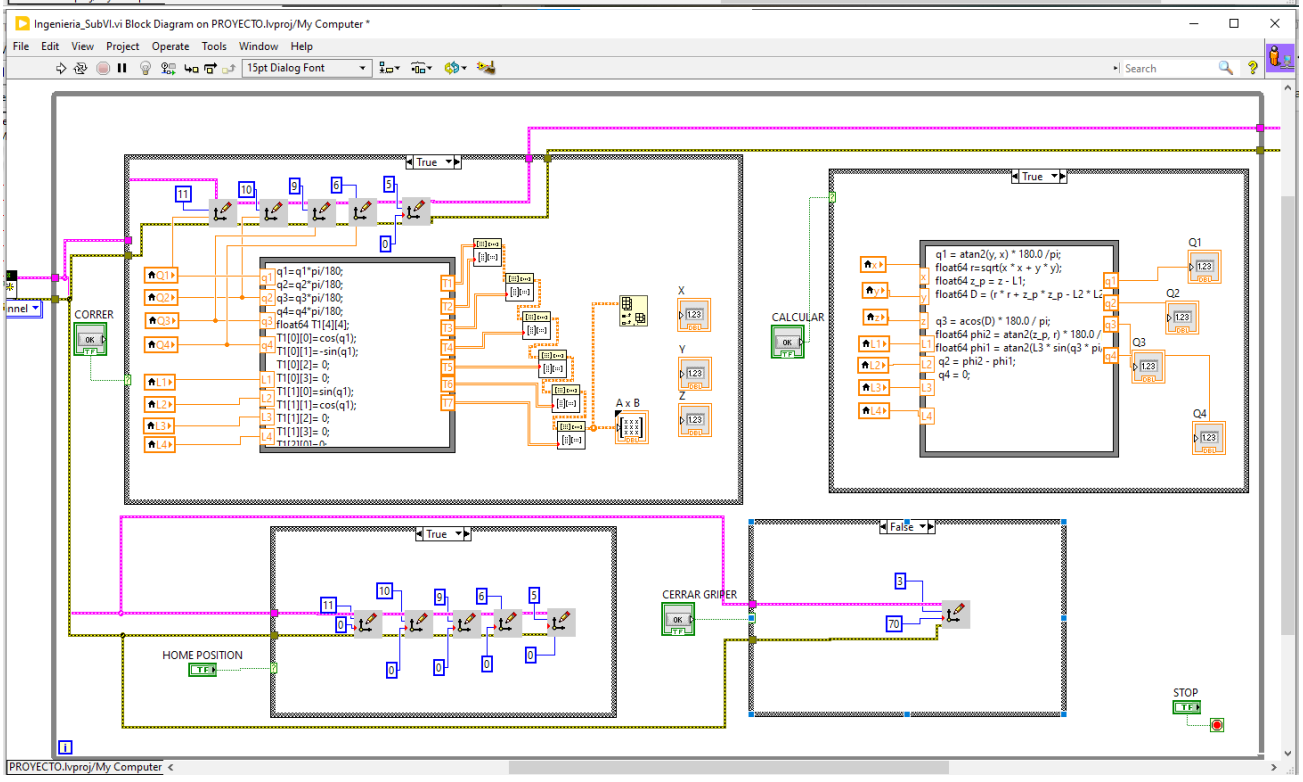

Main VI State Machine
"Exit"
Initial State
Initialize
Initial State
Exit
Default
Error
?!
T
i

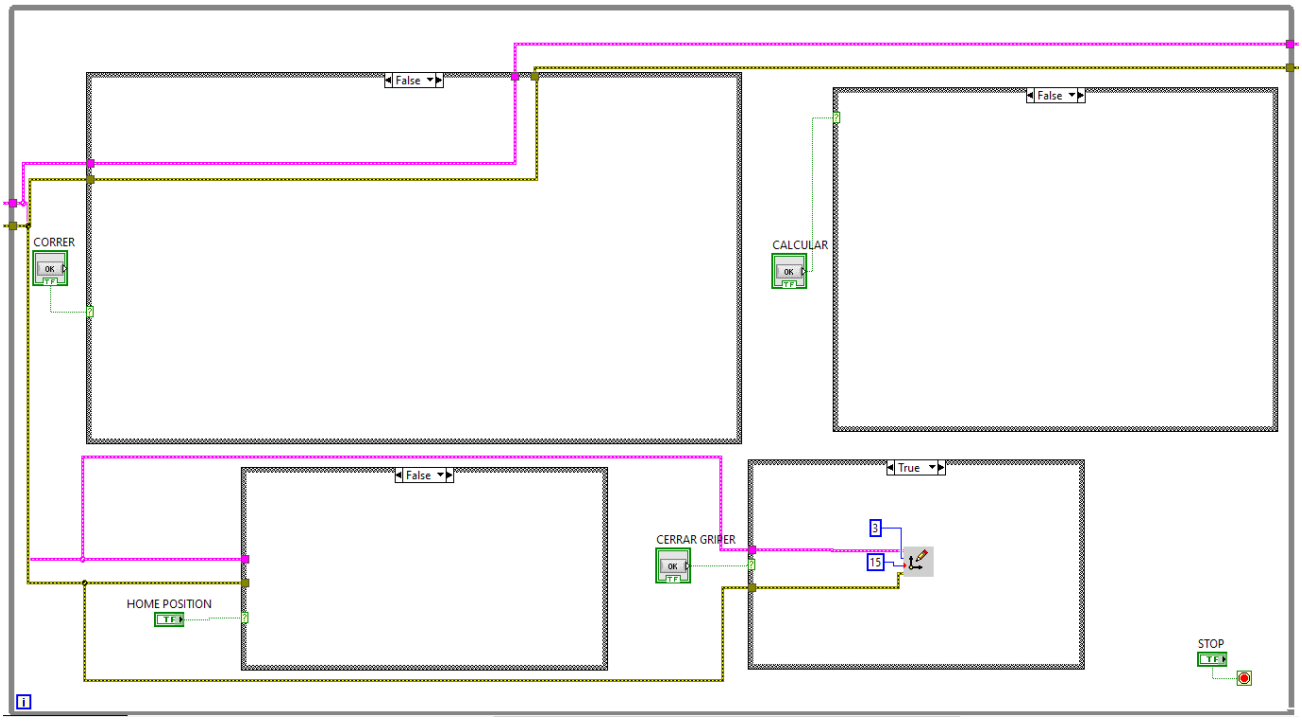

Main VI State Machine
"Initialize"
Initialize - Initialize the application.
Initial State
Initialize
Initial State
Wait for Event
Default
Error
?!
Stop Loop?
F
i

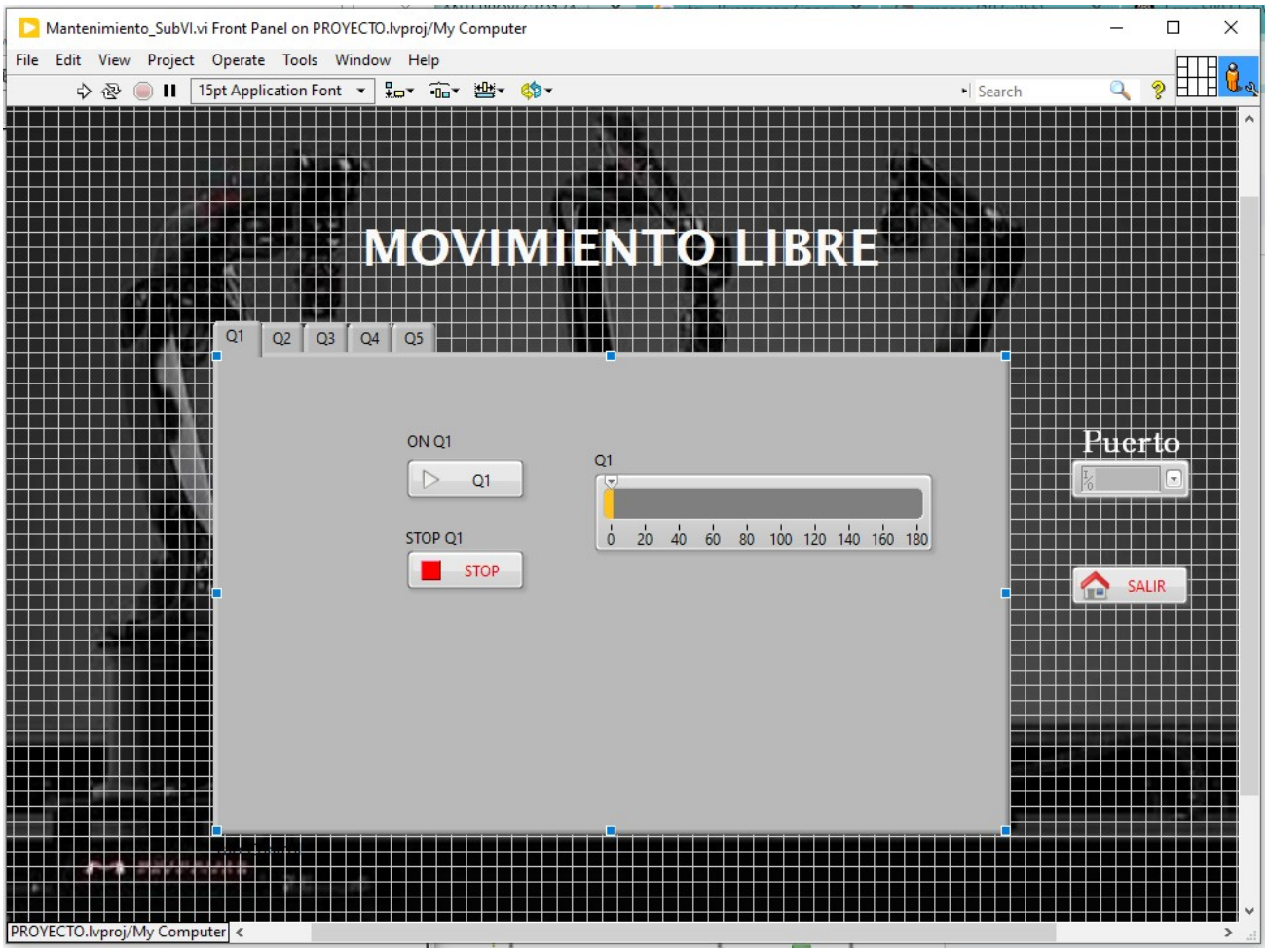

Ingenieria_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
ROBOT DIDACTICO DE 4 DOF
Puerto
Cinematica Directa
L1
L2
L3
L4
Q1
Q2
Q3
Q4
X
Y
Z
CORRER
Cinematica Inversa
x
y
z
Q1
Q2
Q3
Q4
CALCULAR
HOME POSITION
GRIPER
SALIR
PROYECTO.lvproj/My Computer

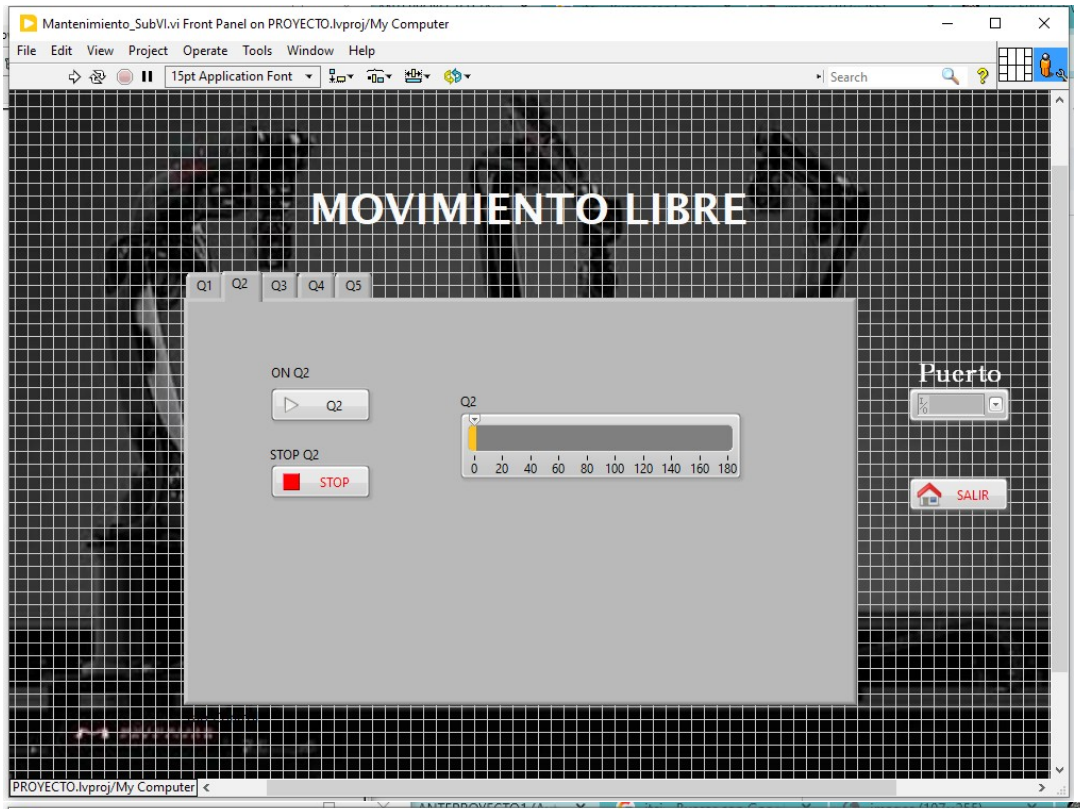

Ingenieria_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Puerto
Error In
Serial
One Channel
CORRER
HOME POSITION
True
PROYECTO.lvproj/My Computer

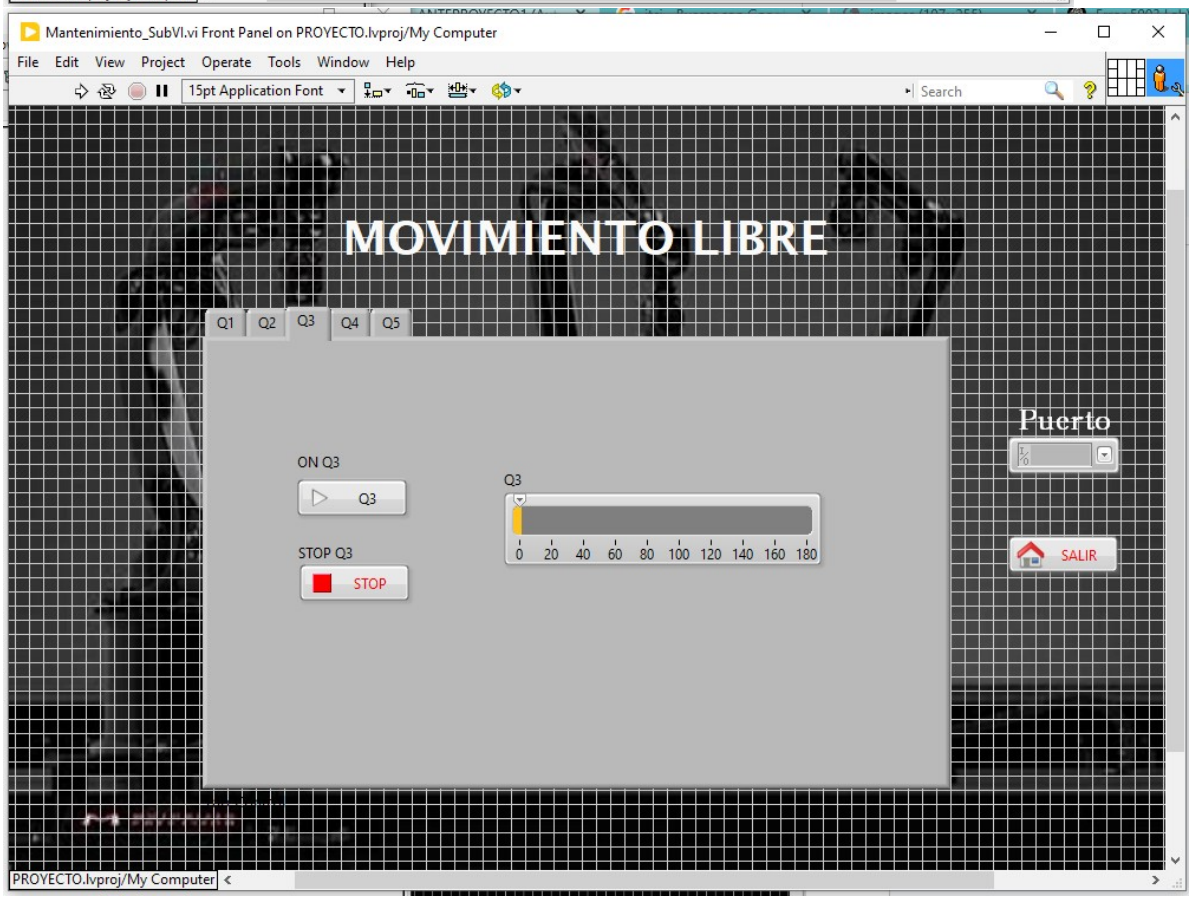

Ingenieria_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
True
CALCULAR
X
Y
Z
A x B
q1 = atan2(y, x) * 180.0 /pi;
float64 r=sqrt(x * x + y * y);
float64 z_p = z - L1;
float64 D = (r * r + z_p * z_p - L2 * L2
q3 = acos(D) * 180.0 / pi;
float64 phi2 = atan2(z_p, r) * 180.0 /
float64 phi1 = atan2(L3 * sin(q3 * pi
q2 = phi2 - phi1;
q4 = 0;
Q1
Q2
Q3
Q4
One Channel
False
CERRAR GRIPER
STOP
PROYECTO.lvproj/My Computer

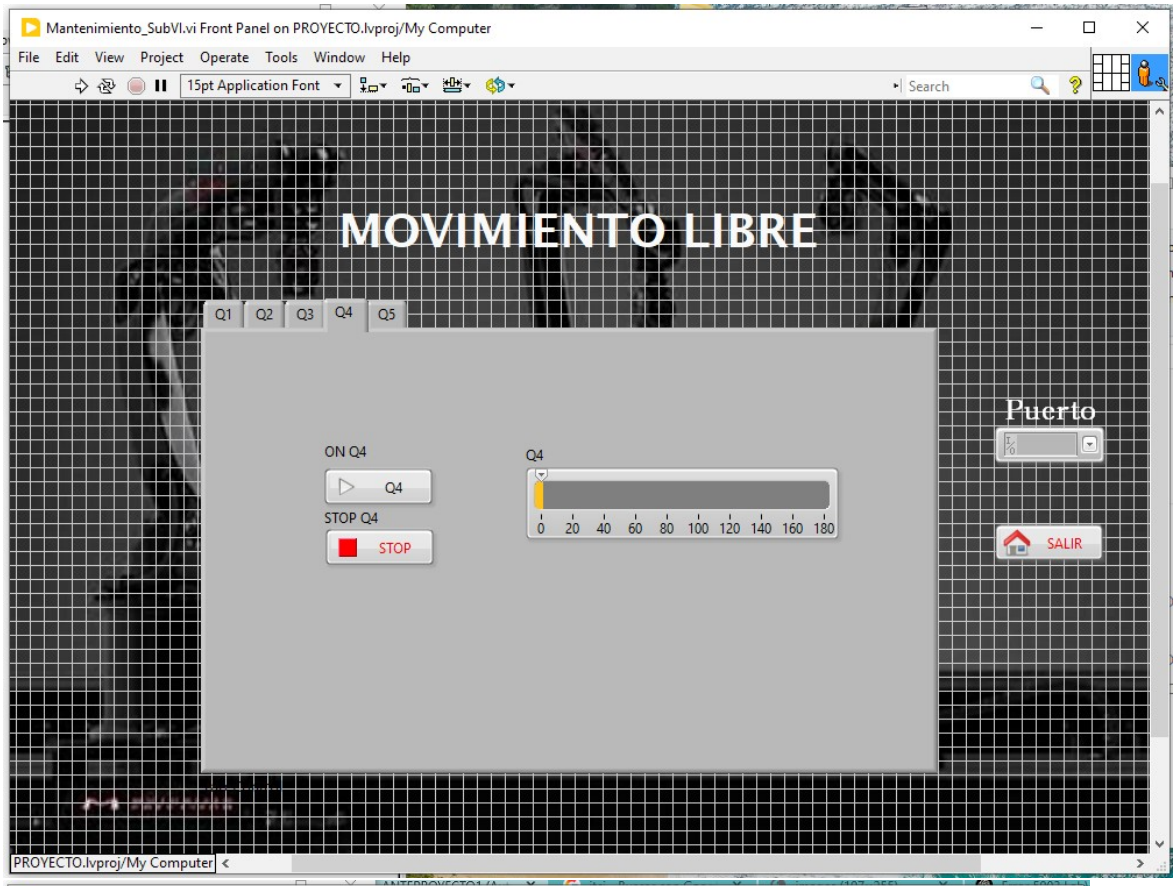

Ingenieria_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
True
CORRER
CALCULAR
HOME POSITION
CERRAR GRIPER
STOP
X
Y
Z
A x B
Q1
Q2
Q3
Q4
PROYECTO.lvproj/My Computer


False
CORRER
False
CALCULAR
False
HOME POSITION
CERRAR GRIPER
True
3
15
STOP

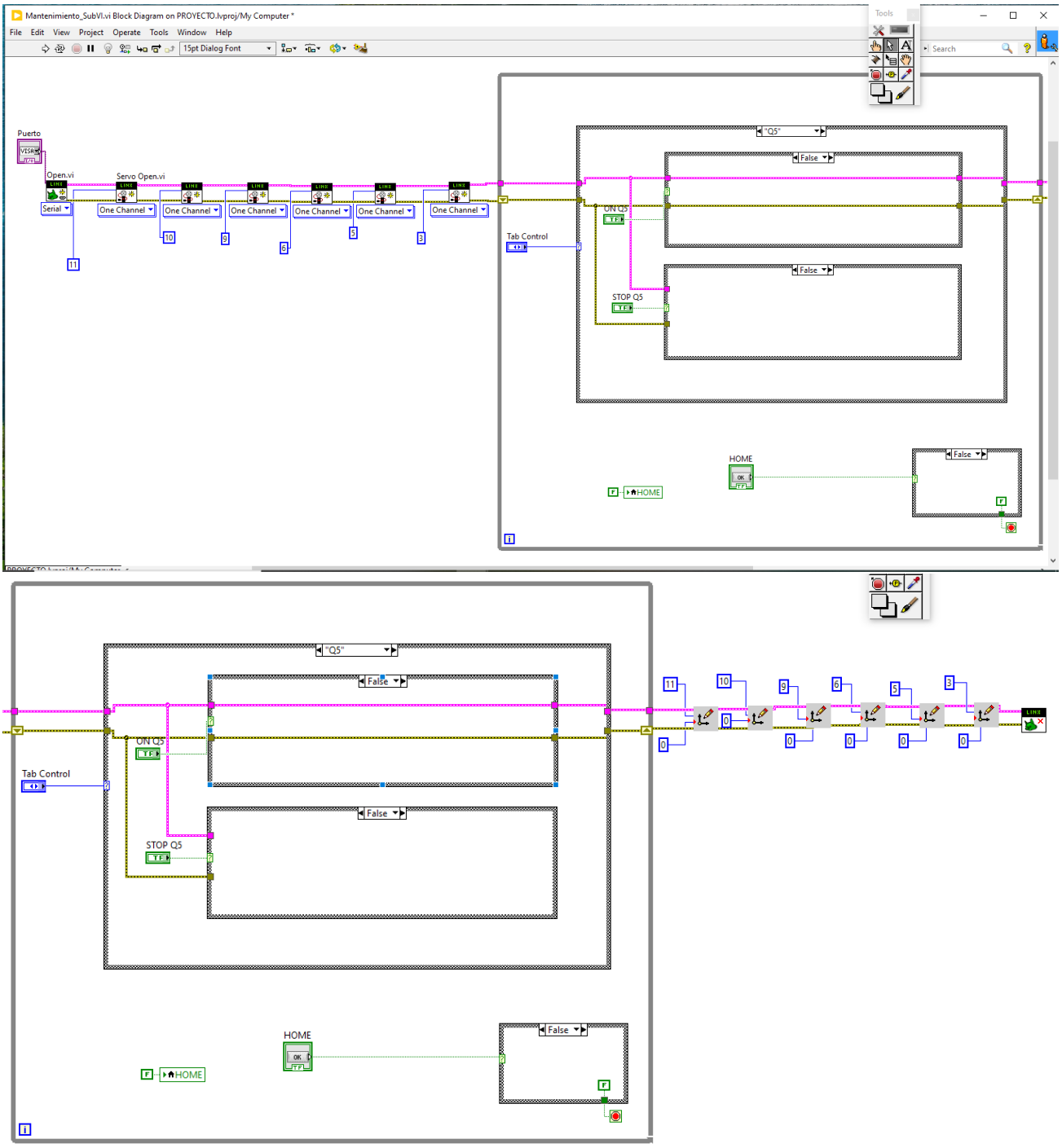

Mantenimiento_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
MOVIMIENTO LIBRE
Q1 Q2 Q3 Q4 Q5
ON Q1
Q1
STOP Q1
STOP
Q1
0 20 40 60 80 100 120 140 160 180
Puerto
SALIR
PROYECTO.lvproj/My Computer

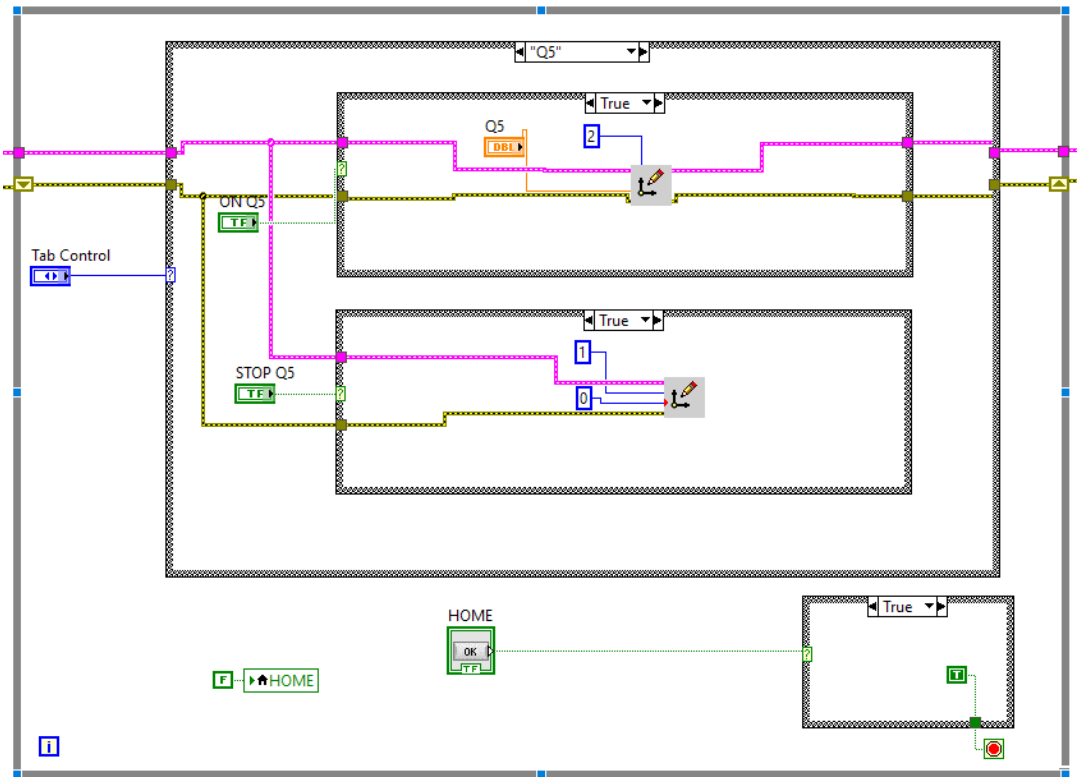

Mantenimiento_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
MOVIMIENTO LIBRE
Q1 Q2 Q3 Q4 Q5
ON Q2
Q2
STOP Q2
STOP
Q2
0 20 40 60 80 100 120 140 160 180
Puerto
SALIR
PROYECTO.lvproj/My Computer

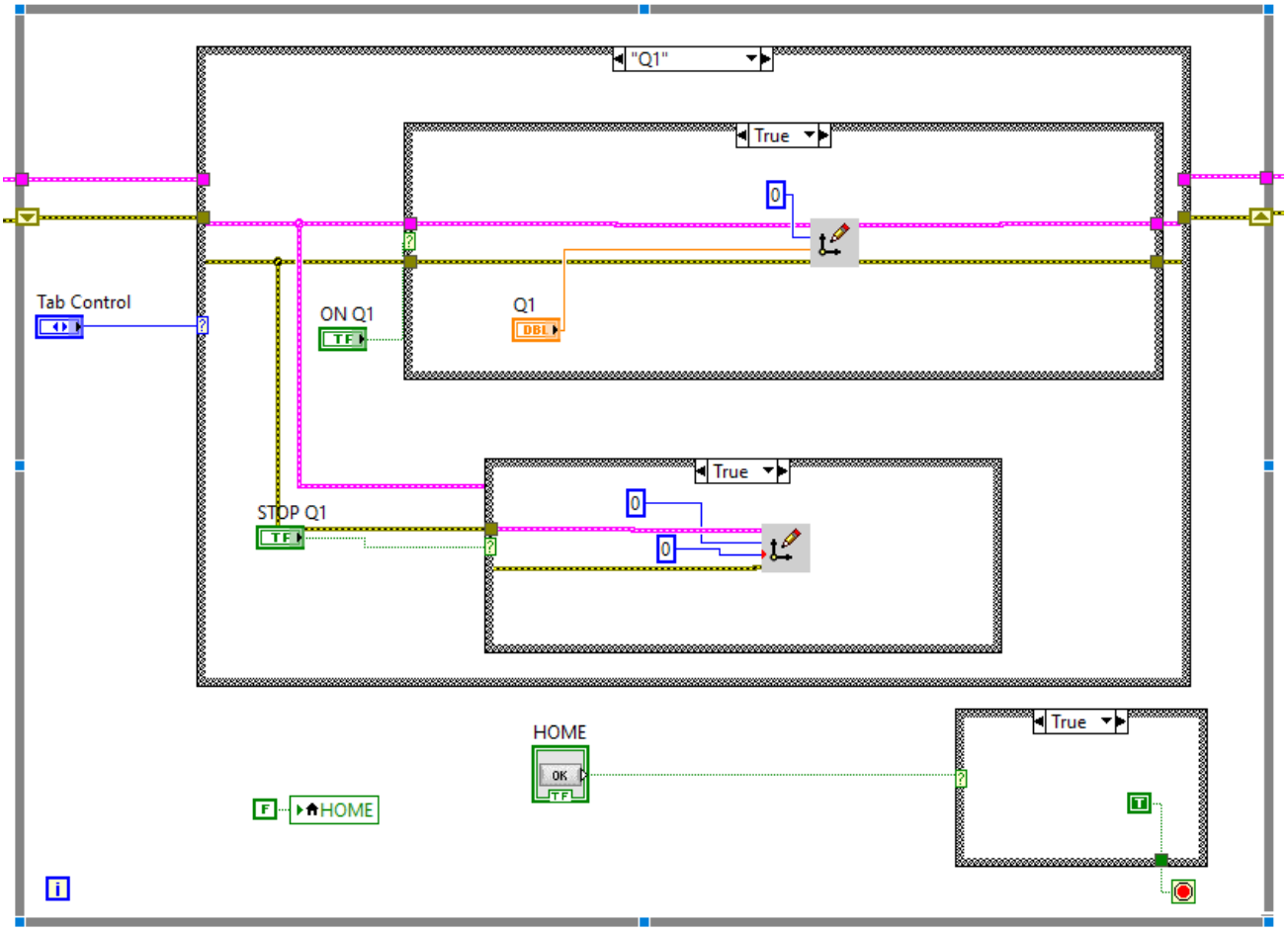

Mantenimiento_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
MOVIMIENTO LIBRE
Q1 Q2 Q3 Q4 Q5
ON Q3
Q3
STOP Q3
STOP
Q3
0 20 40 60 80 100 120 140 160 180
Puerto
SALIR
PROYECTO.lvproj/My Computer

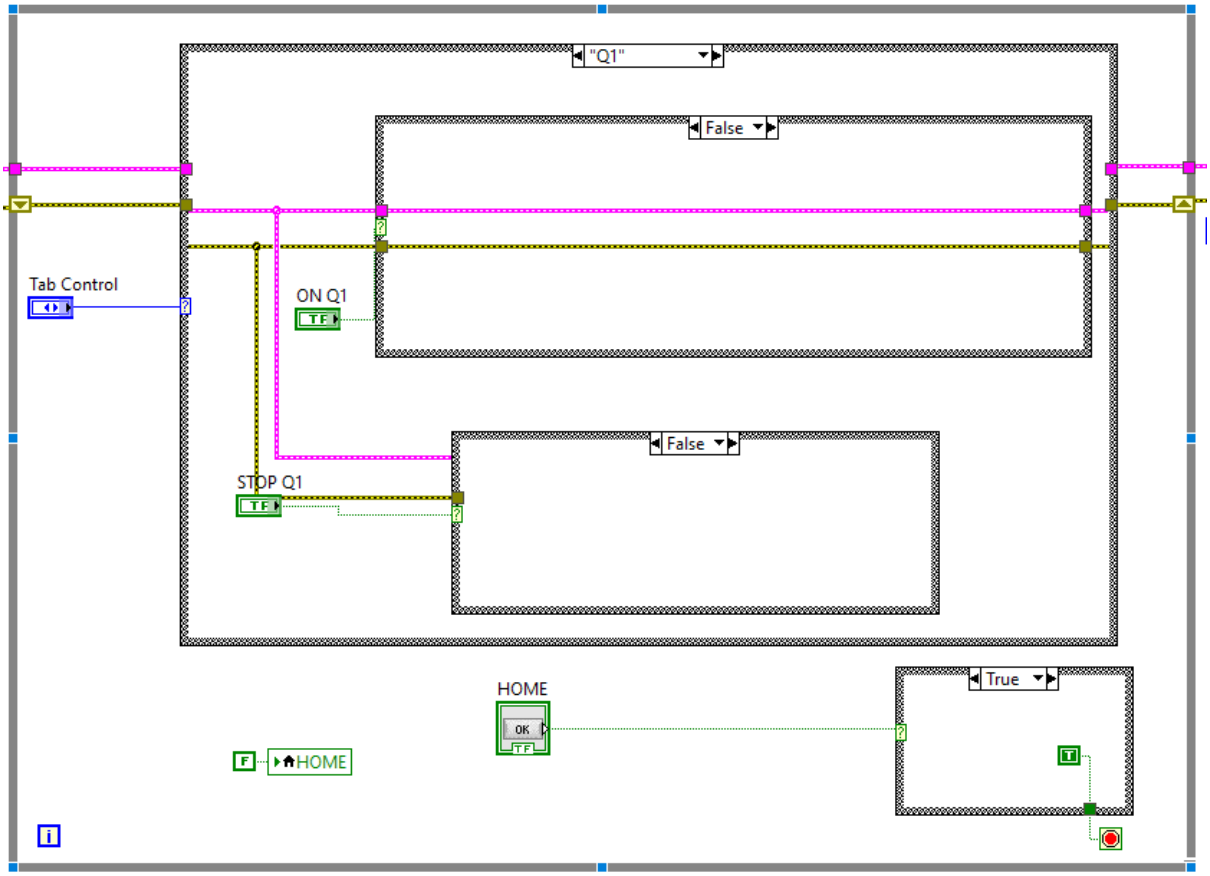

Mantenimiento_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
MOVIMIENTO LIBRE
Q1 Q2 Q3 Q4 Q5
ON Q4
Q4
STOP Q4
STOP
Q4
0 20 40 60 80 100 120 140 160 180
Puerto
SALIR
PROYECTO.lvproj/My Computer

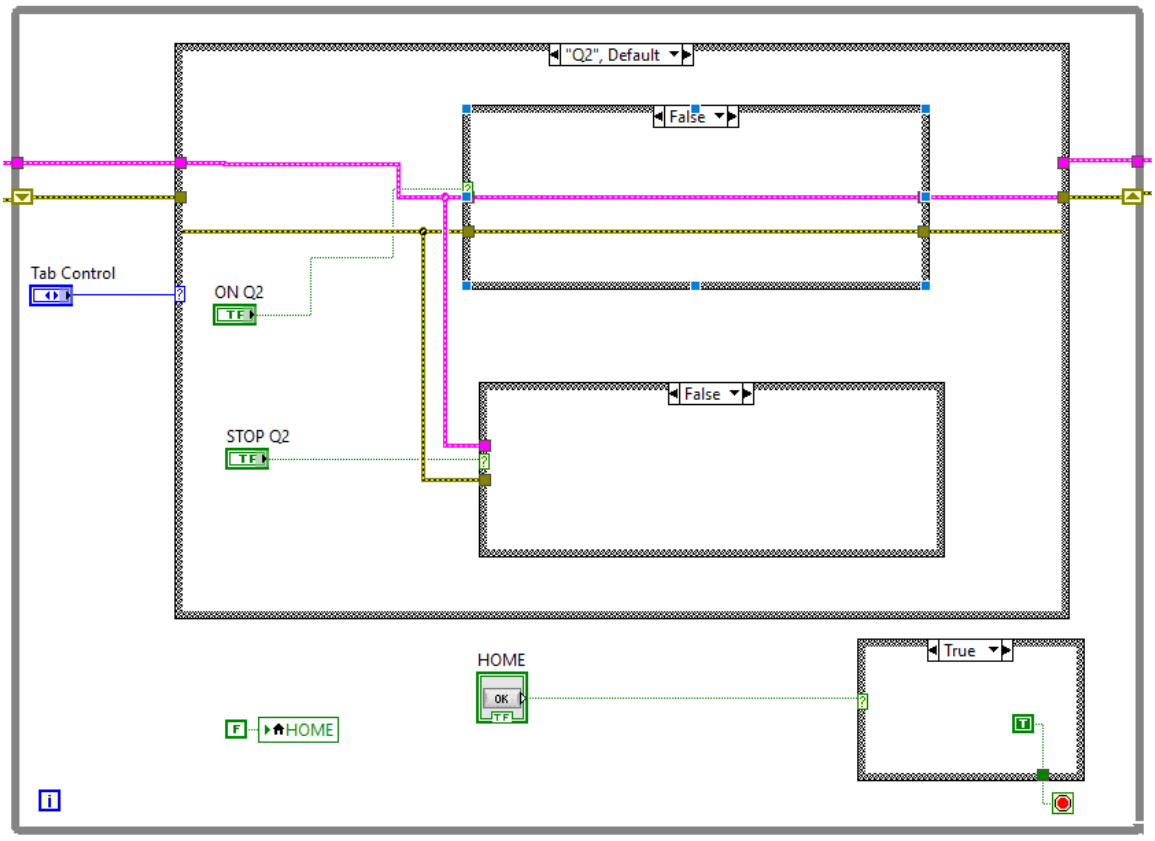
Mantenimiento_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
MOVIMIENTO LIBRE
Q1 Q2 Q3 Q4 Q5
ON Q5
Q5
STOP Q5
STOP
Q5
0 20 40 60 80 100 120 140 160 180
Puerto
SALIR
PROYECTO.lvproj/My Computer

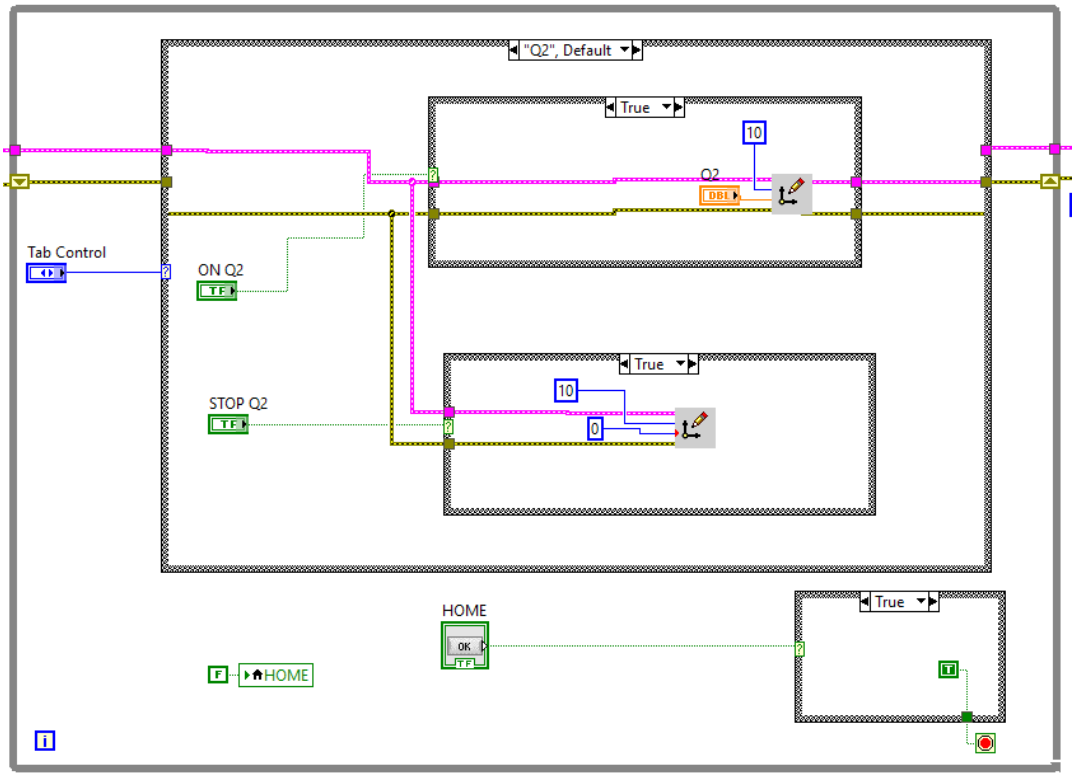

Mantenimiento_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
Tools
Puerto
Open.vi
Servo Open.vi
Serial
One Channel
Tab Control
"Q5"
False
ON Q5
STOP Q5
HOME
HOME
False

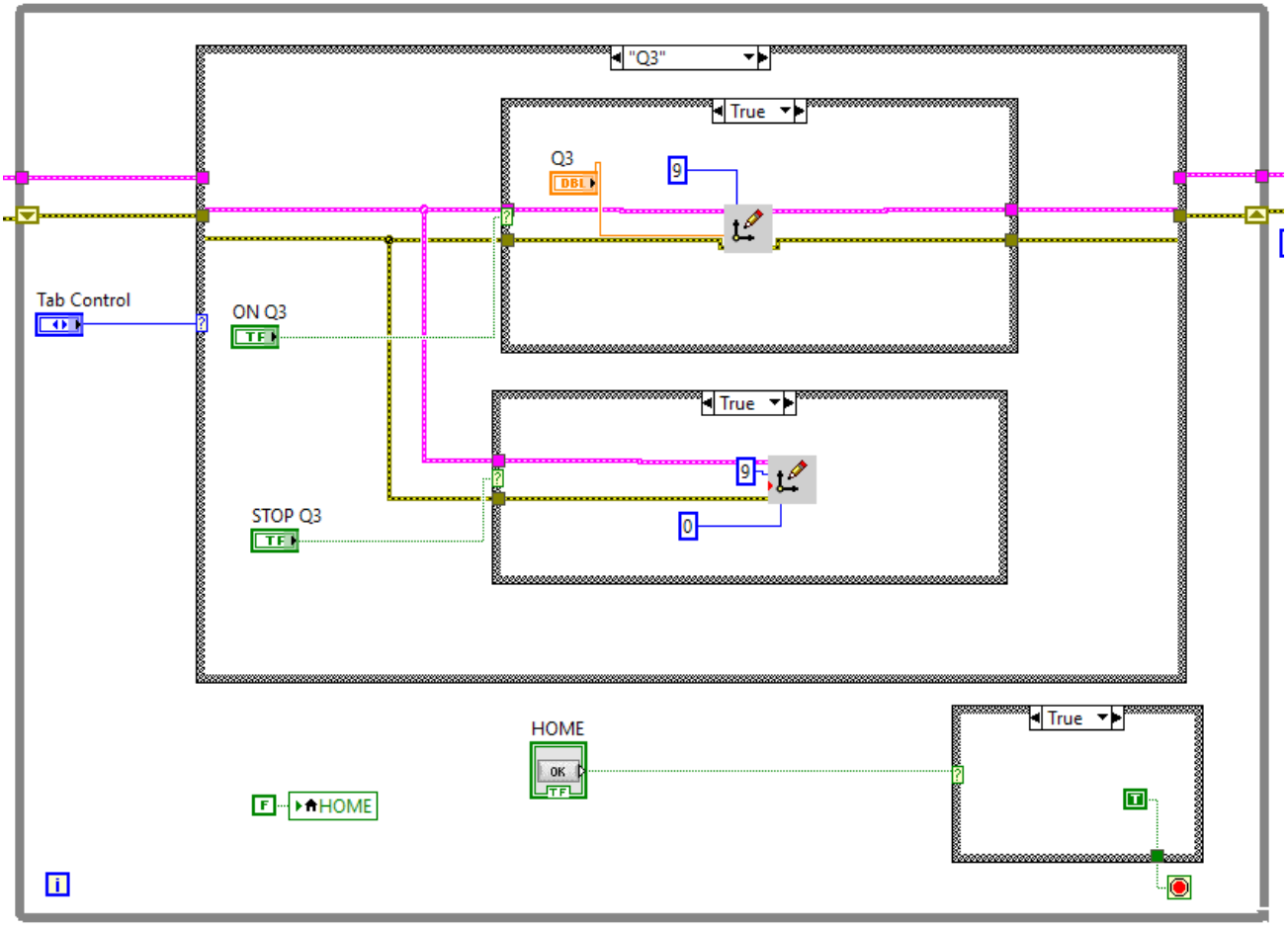

"Q5"
True
Q5
DBL
2
ON Q5
TF
Tab Control
True
1
STOP Q5
TF
0
HOME
OK
TF
F
HOME
True
T
i

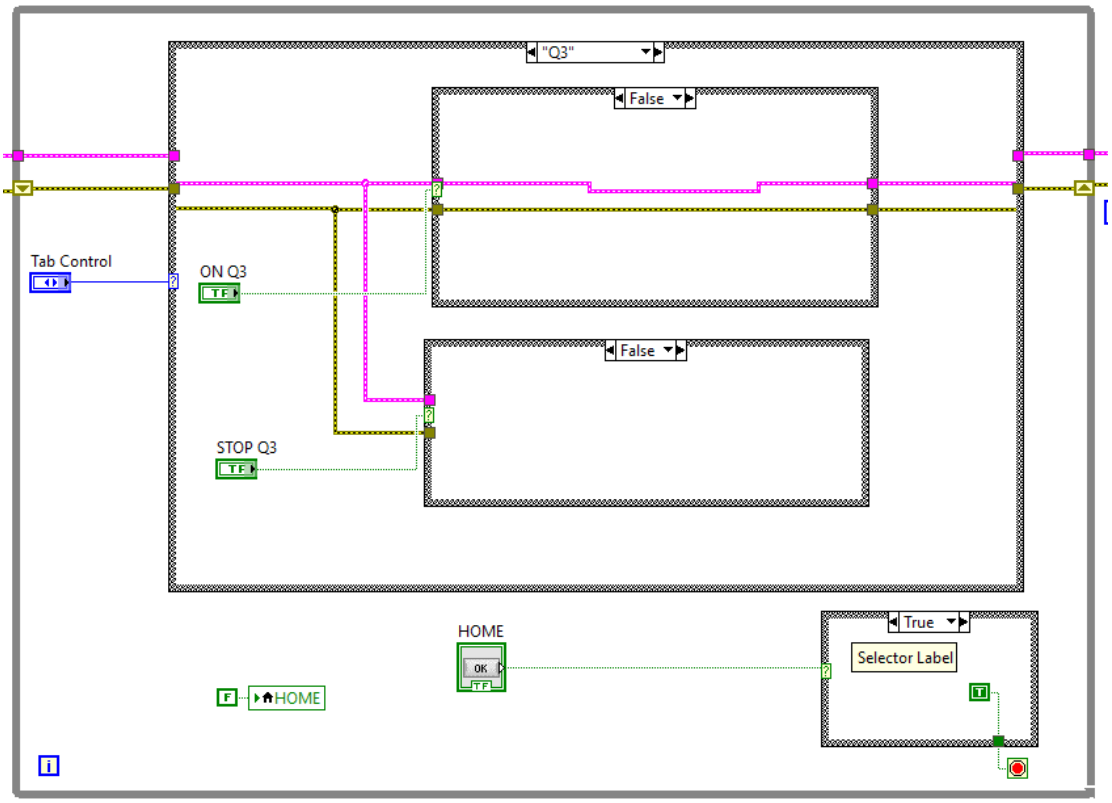

"Q1"
True
0
Tab Control
ON Q1
TF
Q1
DBL
True
STOP Q1
TF
0
0
HOME
OK
TF
F
HOME
True
T
i

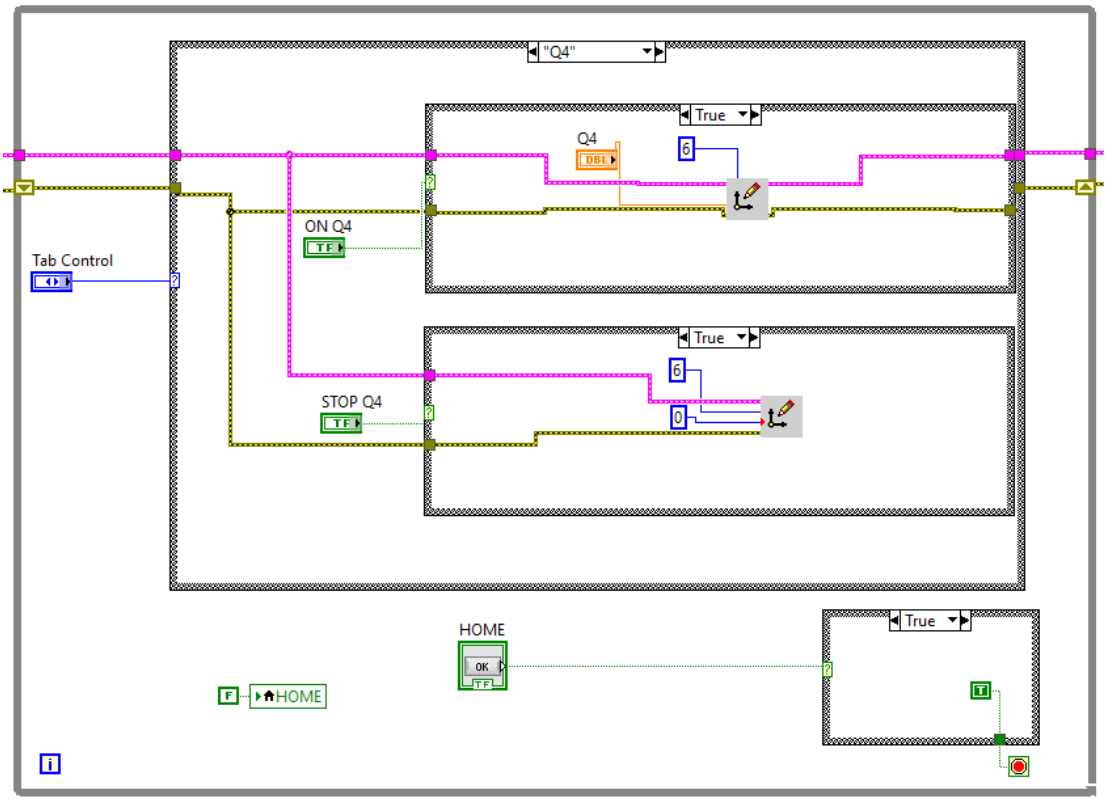

"Q1"
False
Tab Control
ON Q1
False
STOP Q1
HOME
True
HOME

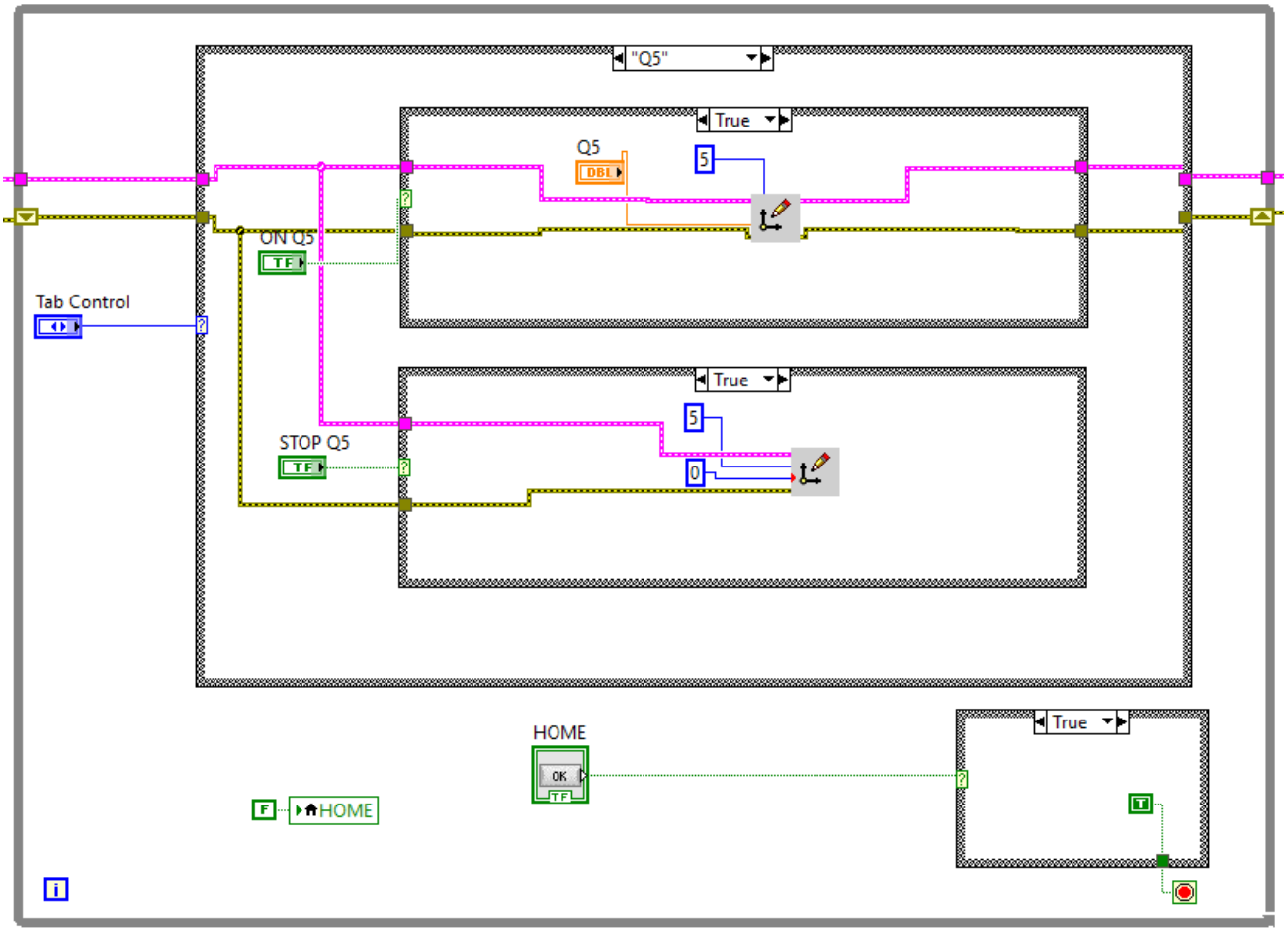

"Q2", Default
False
Tab Control
ON Q2
False
STOP Q2
HOME
True
HOME

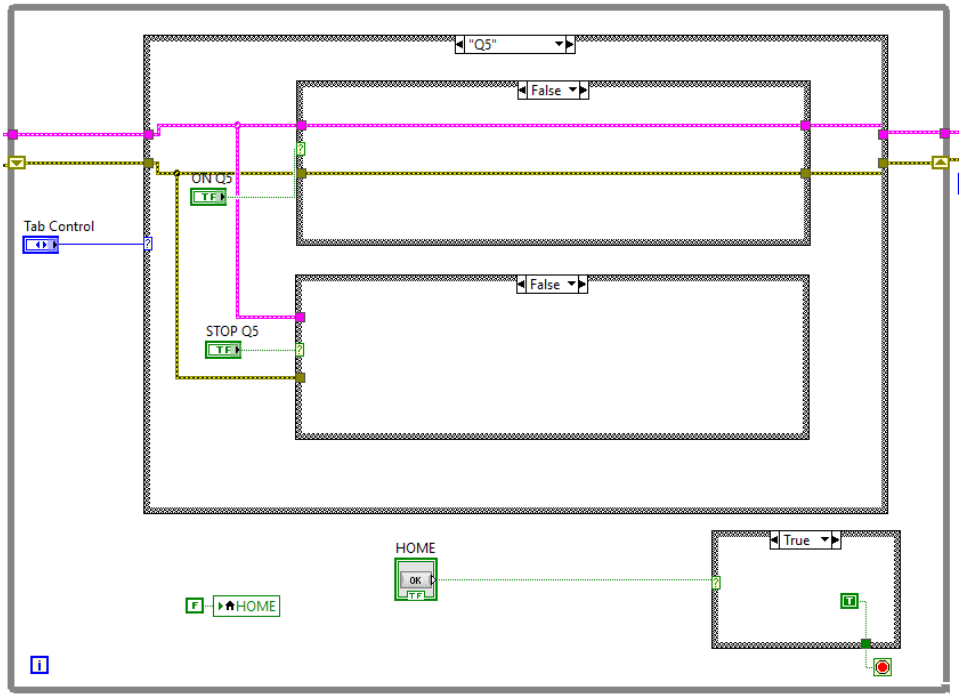

"Q2", Default
True
10
Q2
DBL
Tab Control
ON Q2
TF
True
STOP Q2
10
0
HOME
OK
TF
True
F
HOME
i


"Q3"
True
Q3
DBL
9
Tab Control
ON Q3
TF
True
9
STOP Q3
TF
0
HOME
OK
TF
True
F
HOME
i

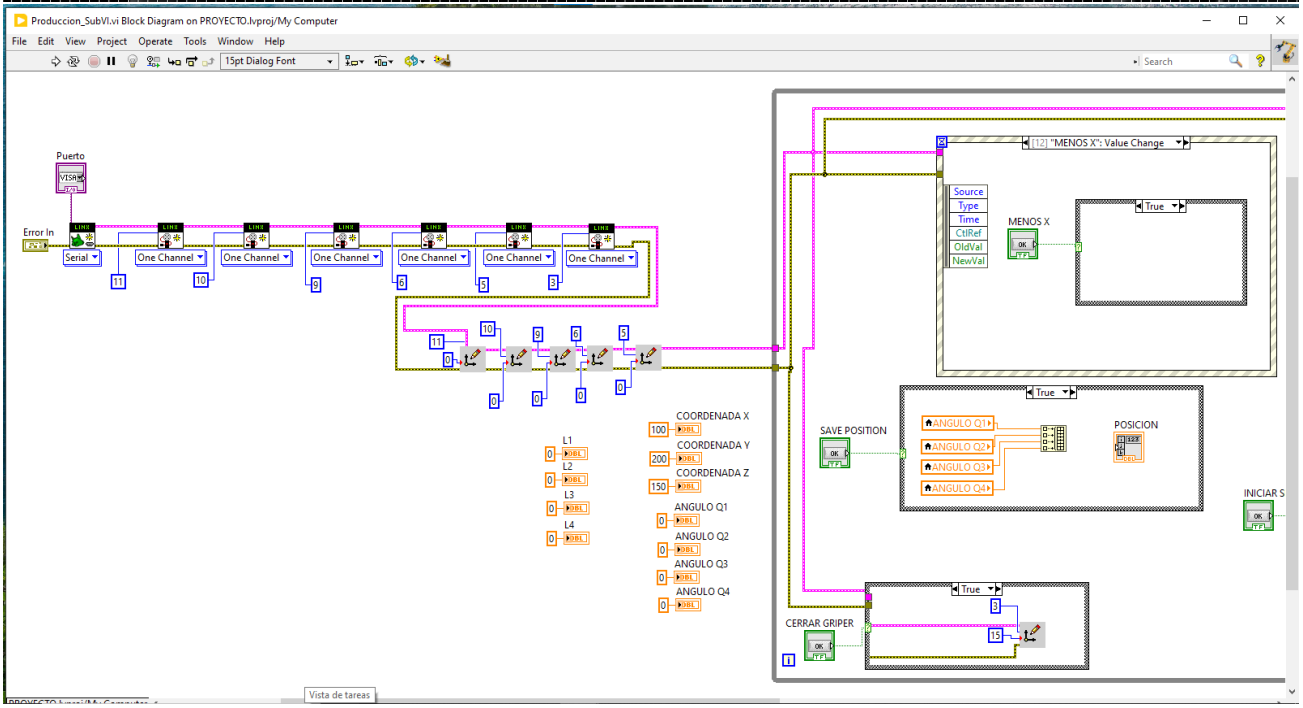

"Q3"
False
False
Tab Control
ON Q3
STOP Q3
HOME
HOME
True
Selector Label

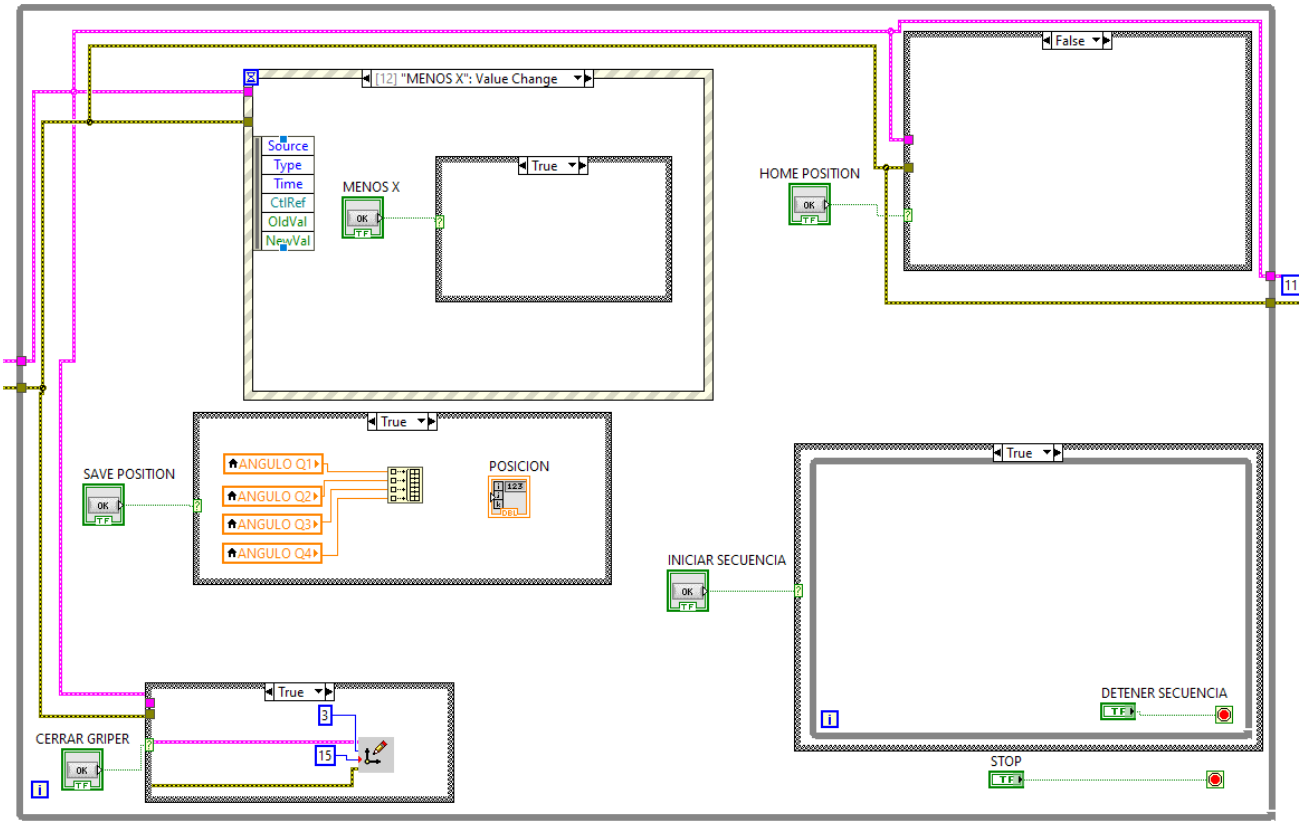

"Q4"
True
True
Q4
DBL
6
6
0
ON Q4
STOP Q4
Tab Control
HOME
HOME
True

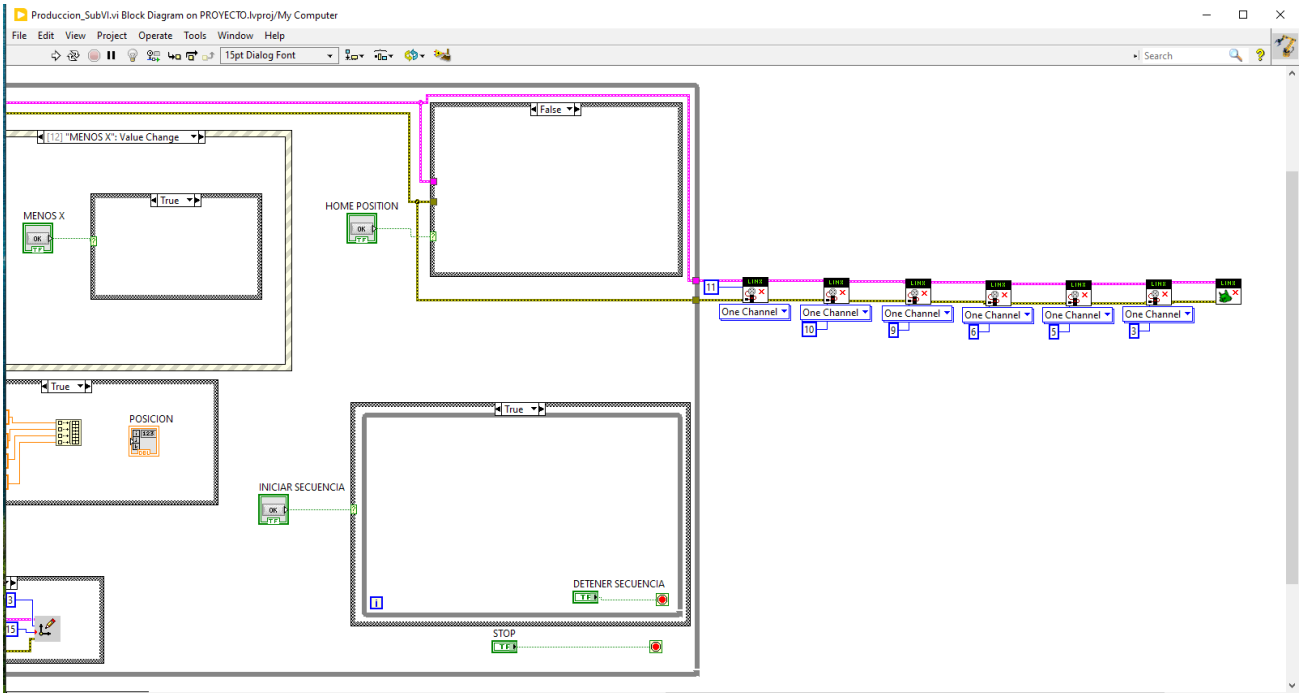

"Q5"
True
Q5
DBL
5
ON Q5
TF
Tab Control
True
5
STOP Q5
TF
0
HOME
OK
TF
True
HOME

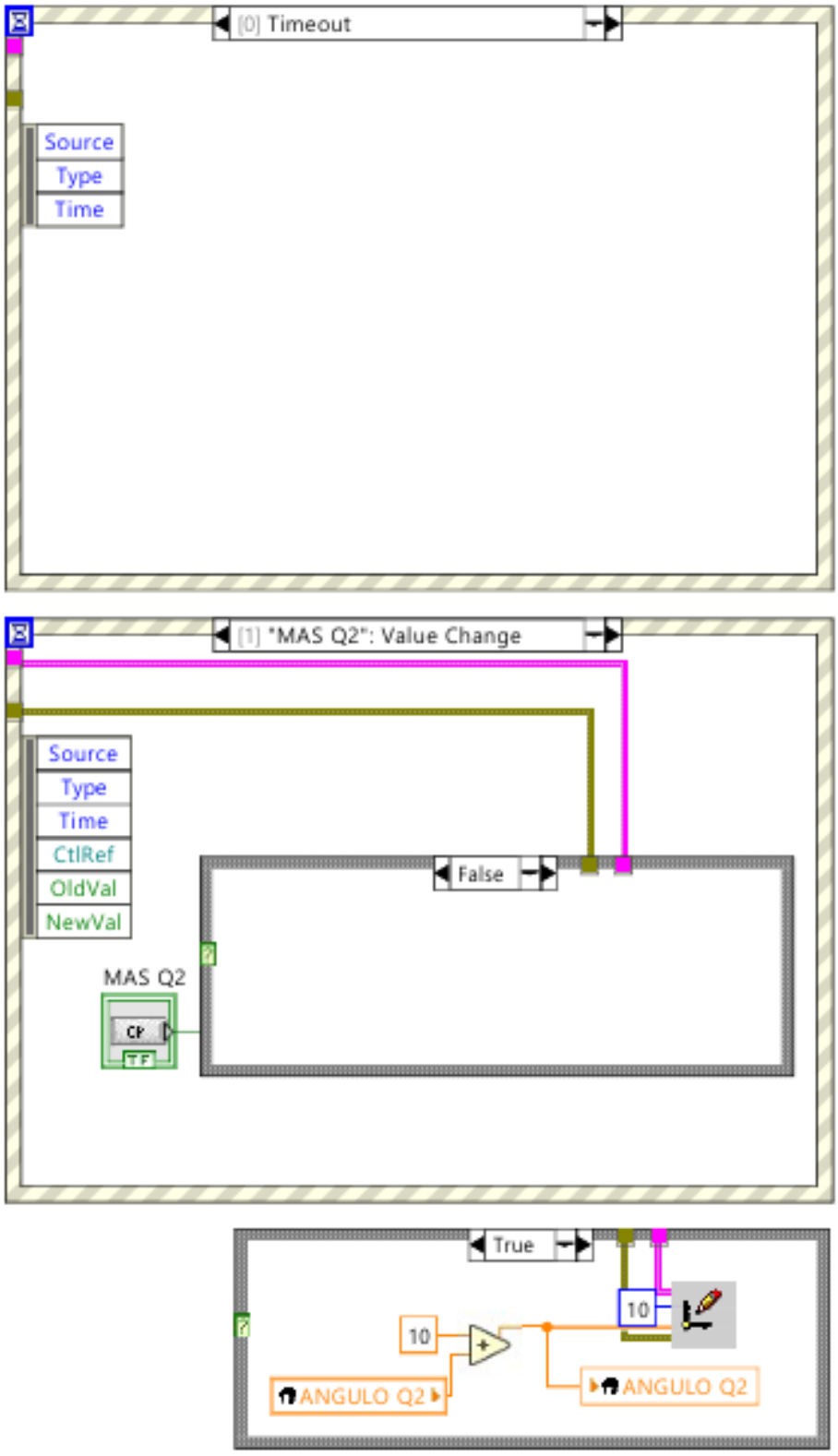

"Q5"
False
ON Q5
TF
Tab Control
False
STOP Q5
TF
HOME
OK
TF
True
HOME

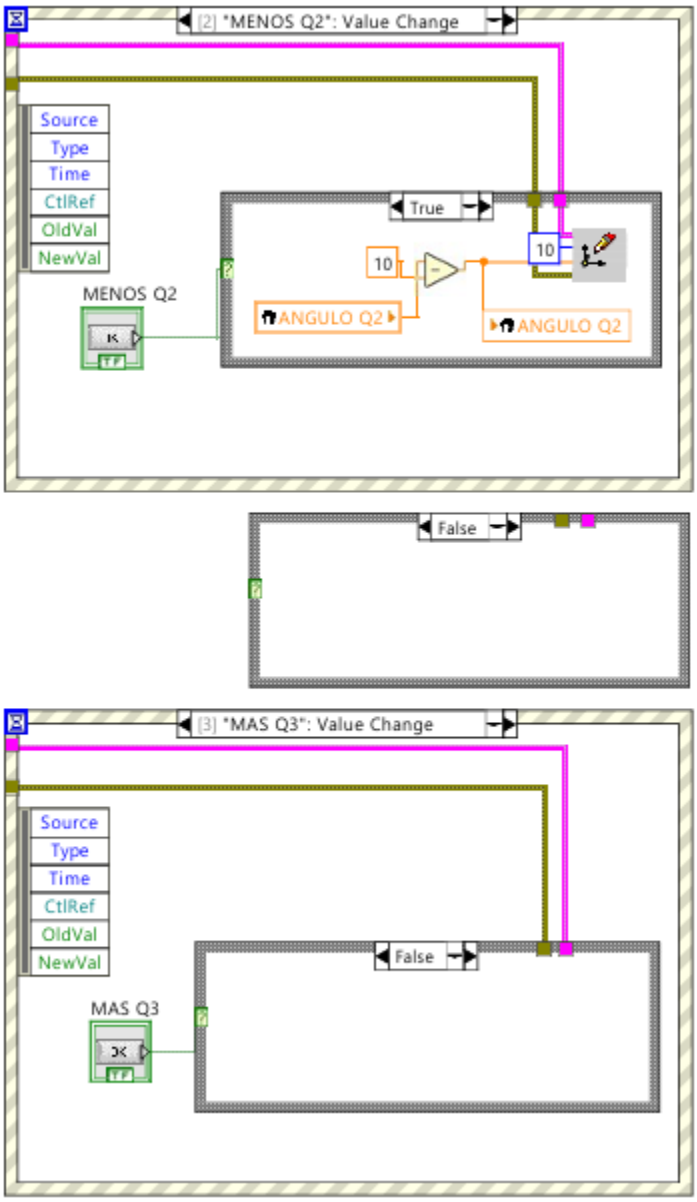
Produccion_SubVI.vi Front Panel on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
24pt Application Font
SECUENCIAS
Puerto
Q1
Q2
Q3
Q4
Z
Y
X
SAVE POSITION
HOME POSITION
GRIPER
INICIAR SECUENCIA
DETENER SECUENCIA
SALIR

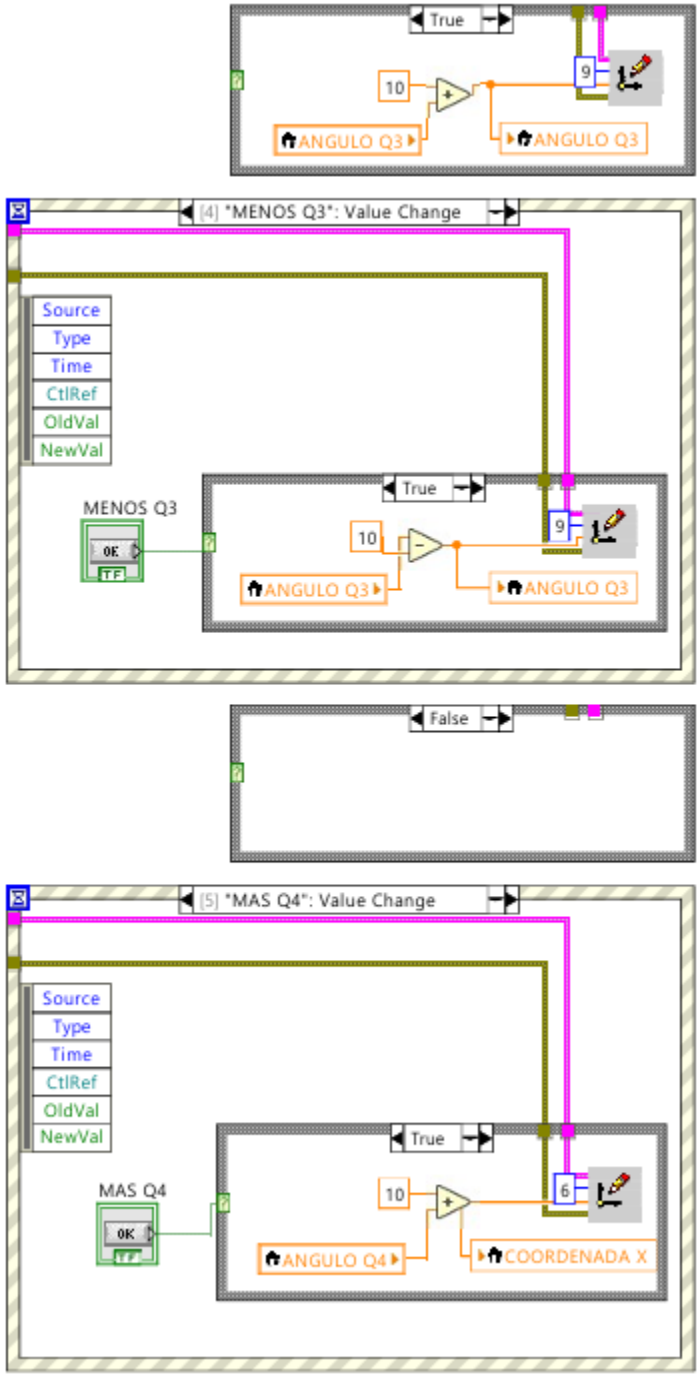

Produccion_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
Puerto
Error In
Serial
One Channel
[12] "MENOS X": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS X
True
SAVE POSITION
ANGULO Q1
ANGULO Q2
ANGULO Q3
ANGULO Q4
POSICION
INICIAR S
CERRAR GRIPER
COORDENADA X
COORDENADA Y
COORDENADA Z
ANGULO Q1
ANGULO Q2
ANGULO Q3
ANGULO Q4
L1
L2
L3
L4
Vista de tareas

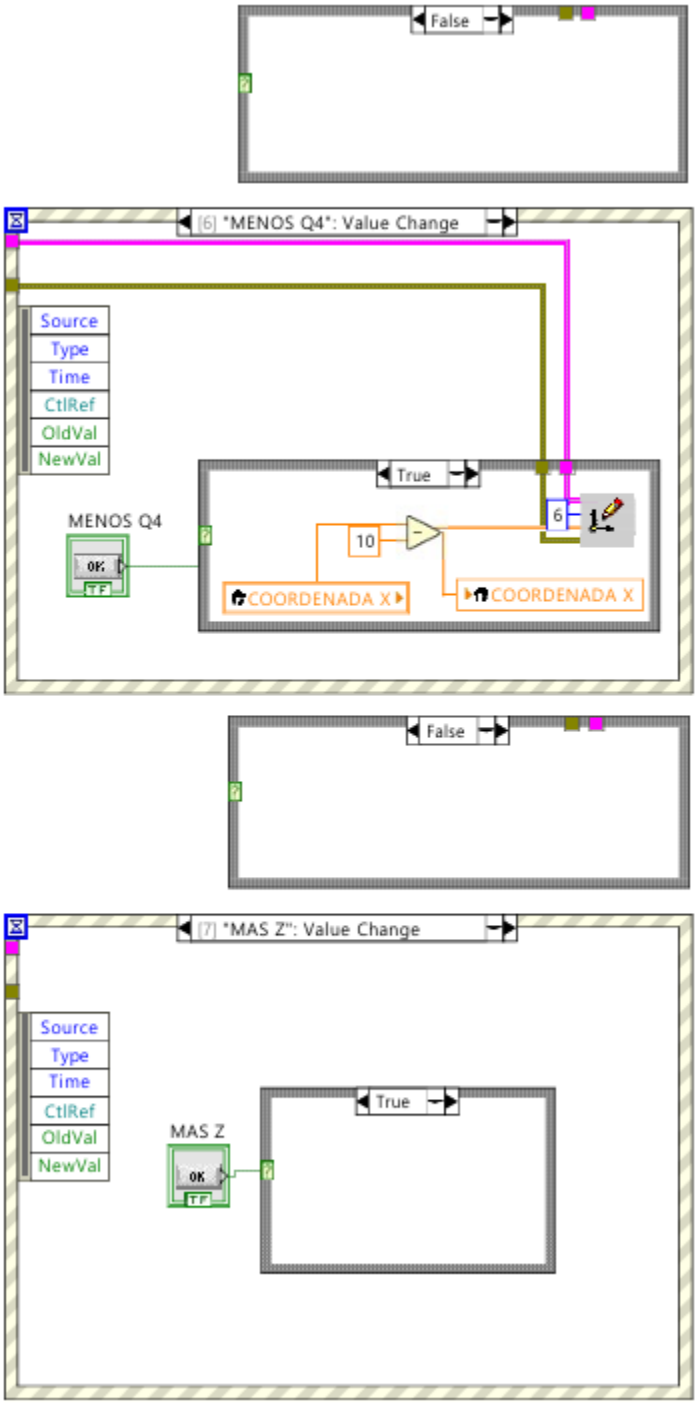

[12] "MENOS X": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS X
True
False
HOME POSITION
SAVE POSITION
ANGULO Q1
ANGULO Q2
ANGULO Q3
ANGULO Q4
POSICION
INICIAR SECUENCIA
DETENER SECUENCIA
CERRAR GRIPER
STOP
3
15
11

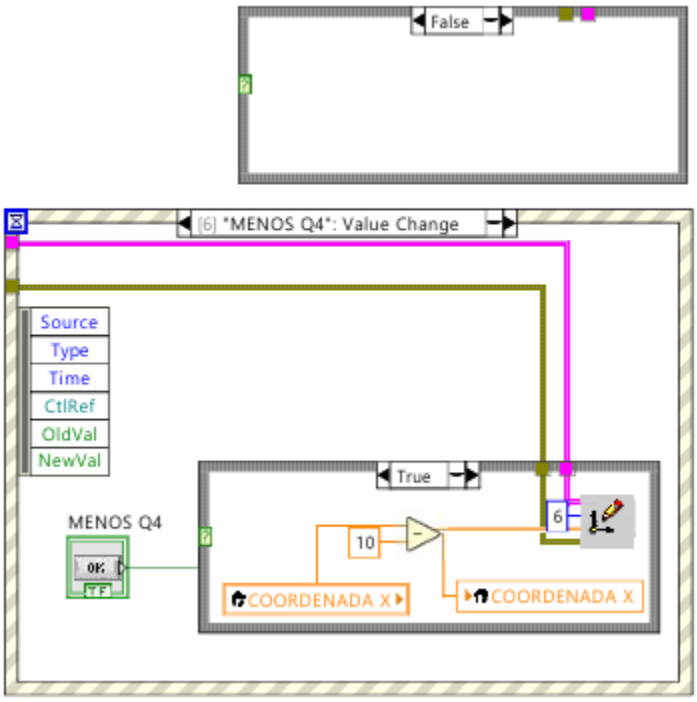

Produccion_SubVI.vi Block Diagram on PROYECTO.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Dialog Font
[12] "MENOS X": Value Change
MENOS X
True
False
HOME POSITION
POSICION
INICIAR SECUENCIA
DETENER SECUENCIA
STOP
One Channel
11
10
9
8
5
3

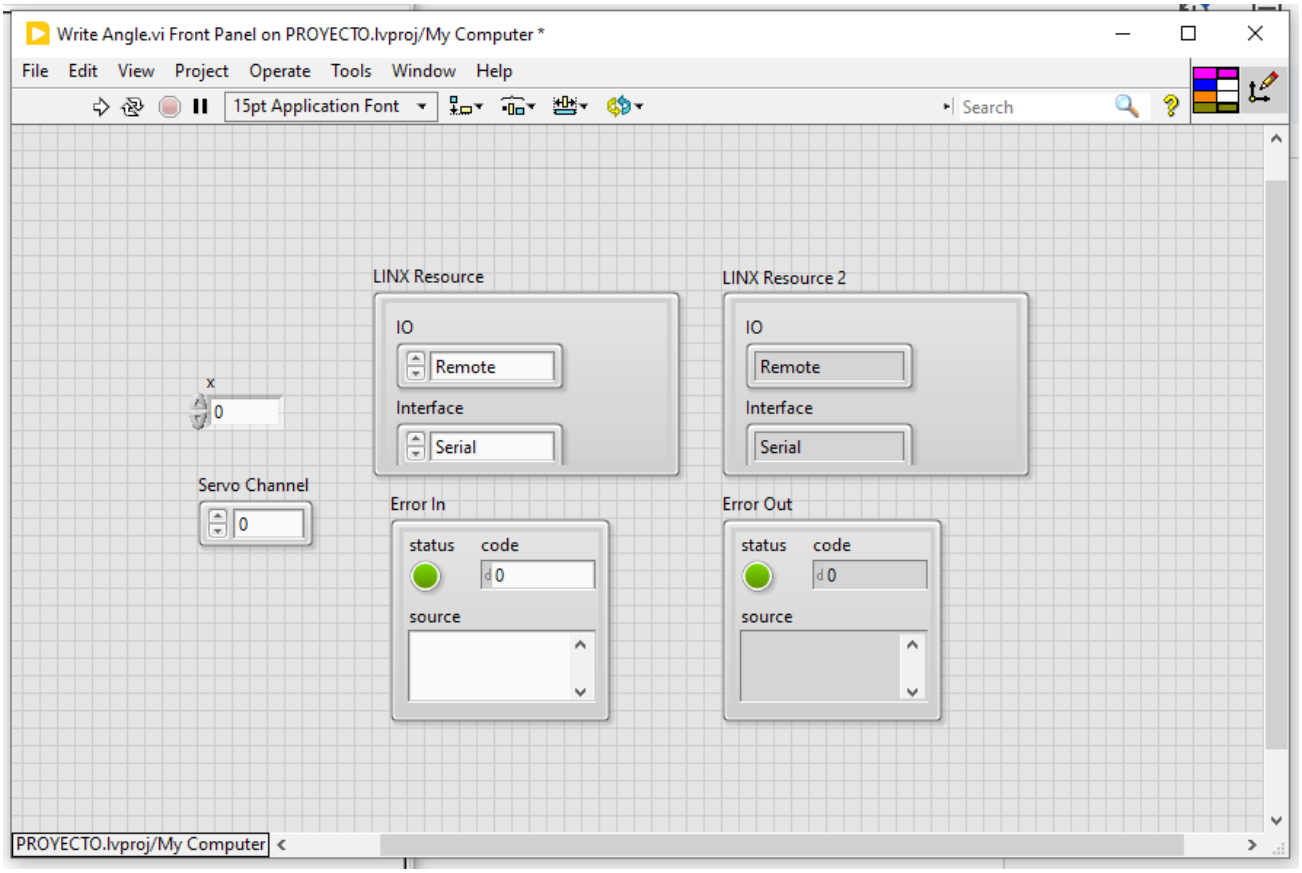

[0] Timeout
Source
Type
Time
[1] "MAS Q2": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
False
MAS Q2
TF
True
10
10
ANGULO Q2
ANGULO Q2

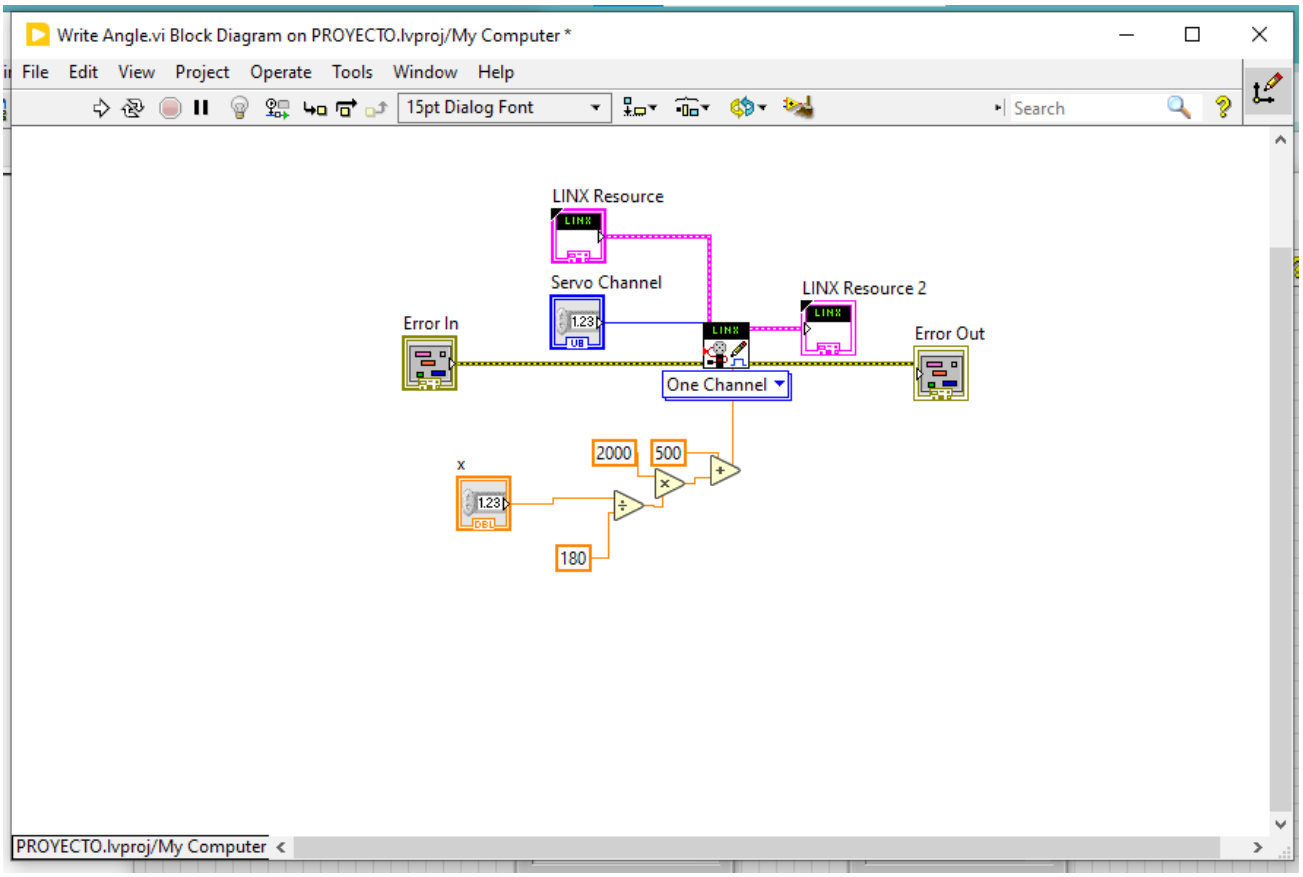

[2] *MENOS Q2*: Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS Q2
True
10
10
ANGULO Q2
ANGULO Q2
False
[3] *MAS Q3*: Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MAS Q3
False


True
10
9
ANGULO Q3
ANGULO Q3
[4] *MENOS Q3*: Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS Q3
True
10
9
ANGULO Q3
ANGULO Q3
False
[5] *MAS Q4*: Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MAS Q4
True
10
6
ANGULO Q4
COORDENADA X


False
[6] "MENOS Q4": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS Q4
True
10
6
COORDENADA X
COORDENADA X
False
[7] "MAS Z": Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MAS Z
True


False
[6] *MENOS Q4*: Value Change
Source
Type
Time
CtlRef
OldVal
NewVal
MENOS Q4
True
10
6
COORDENADA X
COORDENADA X


Write Angle.vi Front Panel on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
LINX Resource
IO
Remote
Interface
Serial
LINX Resource 2
IO
Remote
Interface
Serial
x
0
Servo Channel
0
Error In
status
code
0
source
Error Out
status
code
0
source
PROYECTO.lvproj/My Computer


Write Angle.vi Block Diagram on PROYECTO.lvproj/My Computer *
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
LINX Resource
Servo Channel
LINX Resource 2
Error In
Error Out
One Channel
x
2000
500
180
PROYECTO.lvproj/My Computer

ANEXO 8. Programa del Nodo de Fórmula en Labview para los cálculos de cinemática directa e inversa del robot.

Cinemática directa

```
q1=q1*pi/180;
q2=q2*pi/180;
q3=q3*pi/180;
q4=q4*pi/180;
float64 T1[4][4];
T1[0][0]=cos(q1);
T1[0][1]=-sin(q1);
T1[0][2]= 0;
T1[0][3]= 0;
T1[1][0]=sin(q1);
T1[1][1]=cos(q1);
T1[1][2]= 0;
T1[1][3]= 0;
T1[2][0]=0;
T1[2][1]=0;
T1[2][2]=1;
T1[2][3]= L1;
T1[3][0]=0;
T1[3][1]=0;
T1[3][2]=0;
T1[3][3]=1;
float64 T2[4][4];
T2[0][0]=cos(q2);
T2[0][1]=0;
T2[0][2]=sin(q2);
T2[0][3]=0;
T2[1][0]=0;
T2[1][1]=1;
T2[1][2]= 0;
T2[1][3]= 0;
T2[2][0]=-sin(q2);
T2[2][1]=0;
T2[2][2]=cos(q2);
T2[2][3]=0;
T2[3][0]=0;
T2[3][1]=0;
T2[3][2]=0;
T2[3][3]=1;
float64 T3[4][4];
```

```
T3[0][0] = 1;
T3[0][1] = 0;
T3[0][2] = 0;
T3[0][3] = 0;
T3[1][0] = 0;
T3[1][1] = 1;
T3[1][2] = 0;
T3[1][3] = 0;
T3[2][0] = 0;
T3[2][1] = 0;
T3[2][2] = 1;
T3[2][3] = L2;
T3[3][0] = 0;
T3[3][1] = 0;
T3[3][2] = 0;
T3[3][3] = 1;
float64 T4[4][4];
T4[0][0] = cos(q3);
T4[0][1] = 0;
T4[0][2] = sin(q3);
T4[0][3] = 0;
T4[1][0] = 0;
T4[1][1] = 1;
T4[1][2] = 0;
T4[1][3] = 0;
T4[2][0] = -sin(q3);
T4[2][1] = 0;
T4[2][2] = cos(q3);
T4[2][3] = 0;
T4[3][0] = 0;
T4[3][1] = 0;
T4[3][2] = 0;
T4[3][3] = 1;
float64 T5[4][4];
T5[0][0] = 1;
T5[0][1] = 0;
T5[0][2] = 0;
T5[0][3] = L3;
T5[1][0] = 0;
T5[1][1] = 1;
T5[1][2] = 0;
T5[1][3] = 0;
T5[2][0] = 0;
```

```
T5[2][1] = 0;
T5[2][2] = 1;
T5[2][3] = 0;
T5[3][0] = 0;
T5[3][1] = 0;
T5[3][2] = 0;
T5[3][3] = 1;
float64 T6[4][4];
T6[0][0] = cos(q4);
T6[0][1] = 0;
T6[0][2] = sin(q4);
T6[0][3] = 0;
T6[1][0] = 0;
T6[1][1] = 1;
T6[1][2] = 0;
T6[1][3] = 0;
T6[2][0] = -sin(q4);
T6[2][1] = 0;
T6[2][2] = cos(q4);
T6[2][3] = 0;
T6[3][0] = 0;
T6[3][1] = 0;
T6[3][2] = 0;
T6[3][3] = 1;
float64 T7[4][4];
T7[0][0] = 1;
T7[0][1] = 0;
T7[0][2] = 0;
T7[0][3] = L4;
T7[1][0] = 0;
T7[1][1] = 1;
T7[1][2] = 0;
T7[1][3] = 0;
T7[2][0] = 0;
T7[2][1] = 0;
T7[2][2] = 1;
T7[2][3] = 0;
T7[3][0] = 0;
T7[3][1] = 0;
T7[3][2] = 0;
T7[3][3] = 1;
```

Cinemática inversa:

```
q1 = atan2(y, x) * 180.0 /pi;
float64 r=sqrt(x * x + y * y);
float64 z_p = z - L1;
float64 D = (r * r + z_p * z_p - L2 * L2 - L3 * L3) / (2 * L2 * L3);

q3 = acos(D) * 180.0 / pi;
float64 phi2 = atan2(z_p, r) * 180.0 /pi;
float64 phi1 = atan2(L3 * sin(q3 * pi/ 180.0), L2 + L3 * cos(q3 * pi / 180.0)) *
180.0 / pi;
 q2 = phi2 - phi1;
 q4 = 0;
```